Vanessa Sofia Simões de Ataíde Ramos

Licenciada em Engenharia Informática

A CMMI-compliant Requirements Management and Development Process

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientador: Ana Moreira, Professora Associada, FCT/UNL

Júri:

Presidente:	Prof. Doutor Rodrigo Seromenho Miragaia Rodrigues
Arguente:	Prof. Doutor João Carlos Pascoal de Faria
Vogal:	Prof. Doutor Ana Maria Dinis Moreira

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

A CMMI-compliant Requirements Management and Development Process

Acknowledgments

Throughout this dissertation I was fortunate enough to have the precious contribution of many people and entities. To them, I would like to show my appreciation with a few words of recognized gratitude.

Foremost, I must express my deep gratitude to my supervisors, Ana Maria Moreira and Rui Nunes Gonçalves, for the outstanding guidance and continuous support during this dissertation research and writing. Without them it would, literally, not be possible. I cannot begin to thank my supervisor Ana Moreira, for presenting me with the opportunity to work directly in industry and on one of my favorite topics of software engineering. Her experience, enthusiasm and kindness were a key contribution to finish my academic training with such a feeling of satisfaction. Equally, a special thanks goes to my supervisor Rui Nunes Gonçalves, who I had the privilege to work and learn a lot from. His immense knowledge on the subject of this dissertation and his assistance were a major contribution.

Furthermore, I would like to thank Altran Portugal, for the internship opportunity and for granting me the time and necessary conditions to develop this dissertation. In particular, I would like to thank the Project Manager António Matias, for his assistance, especially during the validation phase.

I would also like to thank Prof. Miguel Goulão, who kindly assisted me with the survey analysis and gave me valuable feedback during the preparation stage. For her assistance and total availability to share her knowledge, I must also thank Denise Bombonatti. Her advice was determinant for the solution proposed in this dissertation.

My sincere thanks also go to the CMMI Portugal community, in particular to Isabel Margarido and to Prof. Fernando Brito Abreu for their precious help and cooperation sharing my survey. Likewise, I would like to extend my appreciation to all the participants of that event that willingly took the time to answer my survey.

For the time spent at the office throughout this year, I must also thank my colleagues at Altran, in particular Sónia Faleiro and João Silva, who made it easier to face every day with motivation. Thank you for your friendship, companionship and help.

Last but not the least, I would like to sincerely thank my family and friends. In particular, my parents, brother and specially my grandparents for always believing in me, even when I didn't, for giving me the opportunity to pursue my dreams and for the unconditional love and support.

To them, I dedicate this dissertation.

Resumo

A Engenharia de Requisitos é uma disciplina essencial para a qualidade do Software. Processos mal definidos para elicitar, analisar, especificar e validar requisitos podem resultar em problemas ou mal-entendidos sobre as necessidades de negócios e âmbito do projeto. Isto leva tipicamente à insatisfação do cliente com a qualidade do produto ou com deslizes nos custos e duração do projeto. Os Modelos de Maturidade permitem a uma organização medir a qualidade dos seus processos e melhorá-los de acordo com uma evolução baseada em níveis. O CMMI (*Capability Maturity Model Integration*) aborda estas questões da engenharia de requisitos, definindo um conjunto de boas práticas para melhoria dos processos. A gestão de requisitos e o desenvolvimento de requisitos são duas áreas de processo incluídas nos modelos de maturidade.

A Altran Portugal é uma empresa de consultoria preocupada com a qualidade do seu software. Em 2012, o departamento Solution Center desenvolveu e aplicou com sucesso um conjunto de processos de acordo com o modelo CMMI-DEV v1.3, o que lhes conferiu uma certificação de nível 2 de maturidade. Para 2015, o objetivo é atingir o nível 3 de maturidade.

Esta dissertação de mestrado é parte integrante deste esforço organizacional, endereçando as áreas de processo da Engenharia de Requisitos. O objetivo principal é contribuir para o desenvolvimento dos processos internos da Altran de acordo com as diretrizes da área de processo de desenvolvimento de requisitos.

Para atingir os objectivos desta dissertação, começámos por definir um método de avaliação baseado no CMMI para ajuizar o nível de conformidade dos processos atuais. Isto permitiu demonstrar o alinhamento da metodologia atual com a área de processo de gestão de requisitos e destacar as melhorias necessárias para a conformidade com a área de processo de desenvolvimento de requisitos do CMMI. Com base no estudo das soluções alternativas para as fragilidades encontradas, foi proposto um novo processo de Gestão e Desenvolvimento de Requisitos, que foi posteriormente validado por meio de três abordagens diferentes.

A principal contribuição desta dissertação é o novo processo desenvolvido para a Altran Portugal. No entanto, considerando que os estudos sobre estes temas não são abundantes na literatura, espera-se também contribuir com evidências úteis para o corpo de conhecimento existente, nomeadamente, através de um survey sobre o CMMI e as tendências da engenharia de requisitos na indústria. Mais importante, esperamos que as melhorias resultantes da implementação do processo proposto minimizem os riscos associados aos requisitos, aumentando o desempenho da Altran e aproximando-os do nível de maturidade desejado.

Palavras chave: CMMI, Engenharia de Requisitos, Desenvolvimento de Requisitos, Gestão de Requisitos, Avaliação de Processos

Abstract

Requirements Engineering has been acknowledged an essential discipline for Software Quality. Poorly-defined processes for eliciting, analyzing, specifying and validating requirements can lead to unclear issues or misunderstandings on business needs and project's scope. These typically result in customers' non-satisfaction with either the products' quality or the increase of the project's budget and duration. Maturity models allow an organization to measure the quality of its processes and improve them according to an evolutionary path based on levels. The Capability Maturity Model Integration (CMMI) addresses the aforementioned Requirements Engineering issues. CMMI defines a set of best practices for process improvement that are divided into several process areas. Requirements Management and Requirements Development are the process areas concerned with Requirements Engineering maturity.

Altran Portugal is a consulting company concerned with the quality of its software. In 2012, the Solution Center department has developed and applied successfully a set of processes aligned with CMMI-DEV v1.3, what granted them a Level 2 maturity certification. For 2015, they defined an organizational goal of addressing CMMI-DEV maturity level 3.

This MSc dissertation is part of this organization effort. In particular, it is concerned with the required process areas that address the activities of Requirements Engineering. Our main goal is to contribute for the development of Altran's internal engineering processes to conform to the guidelines of the Requirements Development process area.

Throughout this dissertation, we started with an evaluation method based on CMMI and conducted a compliance assessment of Altran's current processes. This allowed demonstrating their alignment with the CMMI Requirements Management process area and to highlight the improvements needed to conform to the Requirements Development process area. Based on the study of alternative solutions for the gaps found, we proposed a new Requirements Management and Development process that was later validated using three different approaches.

The main contribution of this dissertation is the new process developed for Altran Portugal. However, given that studies on these topics are not abundant in the literature, we also expect to contribute with useful evidences to the existing body of knowledge with a survey on CMMI and requirements engineering trends. Most importantly, we hope that the implementation of the proposed processes' improvements will minimize the risks of mishandled requirements, increasing Altran's performance and taking them one step further to the desired maturity level.

Keywords: CMMI, Requirements Engineering, Requirements Development, Requirements Management, Process Appraisals

Contents

List of Figures

List of Tables

Abbreviations

CL	Capability Level
CMMI	Capability Maturity Model Integration
CMMI-ACQ	Capability Maturity Model Integration for Acquisition
CMMI-DEV	Capability Maturity Model Integration for Development
CMMI-SVC	Capability Maturity Model Integration for Services
BM	Business Manager
ML	Maturity Level
NFR	Non-Functional Requirement
PIIs	Practice Implementation Indicators
PM	Project Manager
PMBoK	Project Management Book of Knowledge
PMI	Project Management Institute
PMP	Project Management Plan
QA	Quality Attribute
RD	Requirements Development
RE	Requirements Engineering
REQM	Requirements Management
SEI	Software Engineering Institute
SCAMPI	Standard CMMI Appraisal Model for Process Improvement
SG	Specific Goal
SGD	Sistema de Gestão de Delivery
SP	Specific Practice
SPI	Software Process Improvement
UML	Unified Modelling Language
V&V	Verification and Validation

Glossary

Approach	Set of techniques that share a common structure of decomposition of information, regardless of the notation used.
CMMI	The integrated approach of the Software Engineering Institute Maturity Model for systems and software engineering process improvement.
Compliance	The capability of an artifact to adhere to standards, regulations, practices or other formally imposed documents.
Component	A deployable, independent unit of software that is completely defined and accessed through a set of interfaces [1]. It is generally a lower-level component of the product and is integrated with other components to "build" the system [2].
Customer	The party responsible for accepting the product or for authorizing payment. Customers are a subset of stakeholders [2].
Customer Requirements	The result of eliciting, consolidating, and resolving conflicts among the needs, expectations, constraints, and interfaces of the product's relevant stakeholders in a way that is acceptable to the customer [2]. They are user-oriented and represent the problem to solve.
Functional Requirement	A requirement that specifies a function that a system or system component must be able to perform [3].
Interface Requirement	A requirement that specifies an external item with which a system or system component must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction [3].
Product Requirements	A refinement of the Customer Requirements into the developers' language, making implicit requirements into explicit derived requirements. The developer uses product requirements to guide the design and building of the product or service [2].
Maturity Model	Model that describes and determines the state of completeness or perfection (maturity) of certain capabilities. It defines simplified maturity stages which measure the completeness of the analyzed artifacts via different sets of (multi-dimensional) criteria [4].
Method	Systematic process, techniques and heuristics used for the development of an activity.
Methodology	Organized, documented set of rules, practices, techniques and tools that can be repeatable throughout a software process.
Non-Functional Requirement	A quality requirement or a constraint on the system. Specifies how a system is supposed to be, in contrast with functional requirements that define what a system is supposed to do. Examples of NFRs are the security, usability or the performance of a system.
Process	A sequence of activities performed for a given purpose, which transform inputs into outputs.
Process Improvement	Changing a software process with the aim of making it more efficient or improving the quality of its outputs.

Quality Attribute Requirement	A property of a product or service by which its quality will be judged by relevant stakeholders. They are characterizable by some appropriate measure. Quality attributes are non-functional, such as timeliness, throughput, responsiveness, security, modifiability, reliability, and usability. They have a significant influence on the architecture [2]
Requirement	(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2) [3].
Requirements Analysis	The process of studying in detail the elicited requirements, in order to resolve conflicts and document the agreed requirements.
Requirements Development	The process of eliciting, analyzing, specifying and validating the requirements from all the development life cycle phases of a system.
Requirements Elicitation	The process of discovering and capturing the needs, expectations and constraints from relevant Stakeholders.
Requirements Engineering	Branch of Software Engineering that encompasses both Requirements Development and Requirements Management processes, along with all the challenges and deliverables associated to each process' activity.
Requirements Management	The process of handling requirements and their changes, ensuring they are properly documented and their dependencies traceable.
Requirements Specification	A systematically represented collection of detailed requirements, which completely describe the external behavior of a system or component.
Requirements Validation	The process of checking whether the requirements fulfill the stakeholders' needs and expectations.
Software Quality	The capability of a software product to satisfy stated and implied needs when used under specified conditions [5].
Solution Requirement	*See:* Product Requirement.
Stakeholder	A group or individual that is affected by or is in some way accountable for the outcome of an undertaking. Stakeholders may include project or work group members, suppliers, customers, end users, and others [2].
Stakeholder's need	The raw information obtained directly from each stakeholder about the problem to solve.
Traceability	The ability to trace a requirements (1) back to its origins, (2) forward to its implementation in design and code, (3) to requirements it depends on (and vice-versa) [6] .
Technical Requirement	A requirement that refers to the technical aspects that the system must fulfill and that limit the solution space beyond what is necessary for meeting the given functional and quality attribute requirements. For instances restrictions on the technology or the hardware used.
Technique	Application of procedures used to perform an activity.

1. Introduction

Now more than ever, companies and organizations strive for software quality. To survive in this difficult economical environment, companies must improve their processes and tools to deliver products or services that meet customers' expectations at the lowest cost and timeframe.

Software Process Improvement (SPI) approaches, like maturity models, aim at increasing software quality by improving the organizational processes [4]. The Capability Maturity Model Integration (CMMI) has significant impact on project predictability and consistency, promoting customers' satisfaction while reducing cost and time [7].

Critical aspects for product quality are also highly dependent on the quality of the software requirements. Although frequently overlooked, Requirements Engineering bridges the communication gap between customers and developers and can have a major impact on the later phases of the life cycle, where mistakes get harder and more expensive to fix. For that reason, this initial phase of development has been repeatedly acknowledged as essential and a key factor to project success. Therefore, CMMI addresses the Requirements Engineering issues by defining a set of best practices for Requirements Management and Requirements Development.

Given the proven impact of the CMMI maturity model [7] and of a well-defined process of Requirement Engineering on software quality [3, 4], the definition of internal processes that comply with the recommended practices of Requirements Management and Development process areas should be a subject of main concern for the organizations.

This chapter introduces and motivates the work performed in this dissertation. A brief description of the problem to solve and its context are presented in Section 1.1, along with the main goals we aim to achieve, in Section 1.2. In addition, Section 1.3 presents the methodology followed to address the problem, closing with the main contributions in Section 1.4 and the structure of this document in Section 1.5.

1.1. Description and Context

Altran Portugal is a consulting company specialized in services for innovation and technology. They have been trying to improve the quality and value of their software products through the implementation of the CMMI Framework. In 2012, the Solution Center department earned a CMMI maturity level 2 certification for their closed and maintenance projects. For 2015, Altran Portugal aims at addressing CMMI-DEV maturity level 3. To achieve this maturity level, their internal processes will have to be further developed to address a set of required process areas. Among them, there is a Requirements Development process area, which concerns the elicitation, specification, analysis and validation of requirements from all life cycle phases, including functional and non-functional requirements.

The focus of this dissertation will be on this process area, addressed in the context of the Solution Center projects of Altran Portugal. Such a subject interconnects two knowledge fields of Software Engineering: Requirements Engineering and Software Quality. In particular, the Requirements Management (already addressed by Altran) and Requirements Development process areas bring together these knowledge fields by exhibiting how Requirements Engineering is used by the CMMI model as a foundation for improving Software Quality.

1.2. Goals

The purpose of this dissertation is to analyze and improve the current processes used at Altran Portugal, to comply with the recommended practices of the Requirements Development process

area of CMMI-DEV 1.3. This will help them to achieve the desired maturity level 3 certification. Hence, we aim at (i) uncovering a set of non-compliances between Altran's current processes and CMMI Requirements Development process area best practices; (ii) validate the compliance between the current processes and the Requirements Management process area; (iii) updating the processes with solutions fitted to their business needs, to bridge the gaps found — our aim is to build a requirements process that can be employed by the company in the future; (iv) supporting the research on the topic of CMMI-based assessments and development of Requirements Engineering processes compliant with CMMI.

1.3. Work Methodology

To accomplish the goals of this dissertation, we started by studying the main concepts of Requirements Engineering and CMMI, the relationship among these concepts and the main contributions given by other authors on the subject of assessments and development of CMMI compliant RE processes. Additionally, the current situation of Altran's processes was analyzed and an assessment method based on CMMI appraisals was proposed. This assessment was conducted through a direct mapping of the CMMI specific practices to the activities, tools and templates currently adopted at Altran Portugal. The processes' fragilities found in this assessment allowed a better definition of the scope of the problem by exposing four research questions that guided the study of the alternative solutions:

Q1) What are the main techniques used to prioritize requirements?
Q2) How to derive the architecture using Quality Attributes?
Q3) What are the main techniques to analyze the risk of requirements?
Q4) How to define scenarios and operational concepts?

Based on the solutions selected, their internal processes were then revised and updated, creating a new requirements management and development process that bridges the gaps initially found. Given the relationship between process areas, the interfaces with other processes were checked to ensure their consistency. To conclude, the new methodology proposed by this dissertation was validated using three different approaches. First, it was presented to some project managers to gather their feedback and refine the process. Then, the template created was applied to a case study. Finally the core activities were tested in a small pilot project.

1.4. Main contributions

The major contribution is to Altran Portugal. First, their internal processes were evaluated regarding the CMMI Requirements Management and Development best practices. Additionally, their processes were improved according to both the weaknesses and the natural misalignments to CMMI level 3 practices found during the evaluation. It is expected that the adoption of the proposed process will minimize the risks of mishandled requirements, increasing their performance and taking them one step further to the desired maturity level 3.

The dissertation itself is another contribution to the research community, as CMMI-based assessments of processes and best practices in organizations are not topics often addressed. In particular, the Survey conducted during the III CMMI Portugal conference on the relationship between CMMI Maturity and the current RE techniques used in the Portuguese industry, as no other similar study was found till the moment of writing this dissertation.

Finally, the methodology followed and the lessons learned can also contribute to help other practitioners improving their own RE organizational processes.

1.5. Document structure

This document is structured in nine chapters and six appendixes, as follows:

- *Chapter 1 – Introduction:* is this introduction.

- *Chapter 2 – Background:* introduces the basics of this dissertation to clarify the main concepts of Requirements Engineering and CMMI, and how they interrelate.
- *Chapter 3 – Requirements Management and Development in Portugal:* gives a short overview of the state of maturity of some of the Portuguese companies that participated in the CMMI Portugal conference and related RE techniques.
- *Chapter 4 – Requirements Management and Development at Altran:* analyses the processes used by Altran and offers an informal Gap Analysis to identify the areas for improvement in relation to the Requirements Management and Development process areas.
- *Chapter 5 – Investigation of Alternative Solutions:* investigates alternative solutions that can bridge the gaps found in the last chapter and their suitability for the organization.
- *Chapter 6 – New Requirements Management and Development process*: describes the new process proposed for Altran, along with an impact evaluation of the changes in other processes and a new CMMI assessment to demonstrate the improvements in compliance with the Requirements Development process area.
- *Chapter 7 – Process Validation*: validates the new process regarding its acceptance and applicability in the organization using three different approaches.
- *Chapter 8 – Related Work*: overviews of works related to the goal of this dissertation.
- *Chapter 9 – Conclusions and Future Work:* reflects on the work developed during this dissertation, including improvement proposals considered as future work.
- *Appendix A – Relationship between CMMI Levels:* provides a table with the relationship between capability and maturity levels according to CMMI.
- *Appendix B – Survey performed at CMMI Portugal:* illustrates the survey made available to the participants and presents the full analysis of its results.
- *Appendix C – CMMI Gap Analysis of the current process:* exposes the full assessment of Altran's current process, with the rating given to all subpractices of the Requirements Management and Requirements Development process areas.
- *Appendix D – CMMI Gap Analysis of the proposed process:* exposes the full assessment of the proposed process, with the rating given to all subpractices of the Requirements Development process area.
- *Appendix E – Validation Survey (used for Altran's experts):* illustrates the survey made to the Project Managers and presents the full analysis of its results.
- *Appendix F – Case Study Application:* provides the requirements documentation generated from the application of the case study.

2. Background

The purpose of this Chapter is to introduce the basics of this dissertation, particularly to clarify the main concepts of Requirements Engineering, in Section 2.1, and CMMI, in Section 2.2. Additionally, the relationship between such concepts is presented in Section 2.3, concluding with a summary of this Chapter in Section 2.4.

2.1. Requirements Engineering

Over the last two decades, Requirements Engineering (RE) has been a widely researched topic and can be considered a relative mature branch of Software Engineering. Sommerville [8], referred to RE as a term that "*covers all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system*". Other more classical definitions have been provided, such as the one by Zave [10], which emphasizes the value of *"real-world goals*" that drive the project and the importance of "*precise specifications*" and "*their evolution over time and across software families*", acknowledging the fact that requirements inevitable change and specifications can be partially reused.

Regardless of the differences between software development life cycle models, RE is most often related to their initial activity [3, 4], which is usually followed by the design, implementation, testing and maintenance activities. According to Alexander and Stevens [9], this critical first stage is expected to take about 5% of project effort, 25% of calendar time and it requires less human resources than later project stages. However, in reality the time dedicated to requirements is much shorter.

2.1.1. Motivation

The primary concern of RE is to satisfy the client's needs. Hence, the main reason to write requirements is to document a common understanding of what system is to be built, and why it is needed in terms of added value for the business of the client organization. Indeed, requirements engineering is a matter of primary importance due to the major role it plays in the success of software projects [4, 8, 9]. Nuseibeh and Easterbrook [14] recognized the success of a system based on the extent to which it meets its stakeholders needs and expectations. These needs and expectations are expressed in the form of requirements, that are identified and documented by the RE process.

Although requirements have been acknowledged as essential and a key to project success, there are still a number of difficulties inherent to this subject. Over the last decade, the Standish Group has pointed out that incomplete requirements, or unclear business goals, are one the most common reasons for project failure [15]. In [9], Alexander and Stevens state that one of the major problems is that insufficient attention is paid to requirements. The consequences of an insufficient focus on RE are well known and agreed in the literature as systems that become over budget, don't meet customer's needs and take longer to develop than initially planned [3, 4, 9]. Since the requirements drive everything that happens later in the project, these consequences can easily snowball to later stages of the development cycle.

Thus, a well-defined process of RE could increase software quality and productivity, therefore saving time, money and effort [13]. RE helps to set the scope for all the subsequent work and bridges the communication gap between the development team and the customer by reaching an agreed view of the system between different groups of stakeholders who may never meet [9].

RE is an intrinsically social activity. Business stakeholders are the most important requirements sources, and humans are social by nature. This strongly influences the RE process, due to the communication problems between business stakeholders and the development team: the process is collaborative, but most often it is simultaneously competitive. Consensus needs to be

achieved and it is up to requirements engineers to deal with these social aspects of human behavior.

2.1.2. Requirements basics

The IEEE Standard Glossary for Software Engineering [3] traditionally defined requirement as *"(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2)"*. However, this definition can be simplified into: "*A requirement is an externally observable characteristic of the system-of-interest*" [16]. Or simply, a requirement is a characteristic or a certain quality related to a system's functionality.

It has been argued that requirements should be clearly separated (advisedly in different documents) according to its different levels of abstraction, to enable an effective communication between readers with different viewpoints [3, 5]:

- **The user requirements** are *user-oriented* as they represent the user's needs, expectations and constraints. They are statements written in natural language and supported by simple diagrams.
- **The system requirements** (or functional specifications) are *developer-oriented,* since they provide a more formal and detailed description of the problem. They represent what the system should do to meet the user's needs and define exactly what will be implemented.

In the context of Requirements Engineering, software requirements are typically classified as *functional* and *non-functional*. A functional requirement describes what the system should do, while a non-functional requirement is related to how well it actually does it. Although the answer to 'how these requirements are to be attained' is often mistaken for a requirement, it is in fact related to a design decision and should be left to later development stages [9].

Functional requirements (FR) are described as '*statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations*' [1].

Non-functional requirements (NFR) lack a universal definition [17]. They can be seen as a quality or a constraint on functionality or on the system. NFRs have been also referred to as the "-ilities" to express quality attributes such as Security or Usability, but in fact are broader than that. Quality attributes are a subset of NFRs, they also include constraints related to time, cost, the development process, the implementation, the standards or policies used or any other not concerning system's functionality [1]. Lamsweerde breaks down the NFRs in quality of service, restrictions of architecture and development or law enforcement [18]. Quality attributes also have themselves several schemes of classification, which, for instances, can be found in the catalogues proposed by Chung [19].

2.1.3. Requirements Engineering processes

Processes are essential to simplify complex interactions and allow knowledge to be reused. In [8], Sommerville defines requirements engineering process as a "*design process which transforms inputs into outputs"* (see Figure 2.1). In other words, it is an organized set of activities, which require creativity, engineering, judgment and interaction between actors with different roles and expertise.

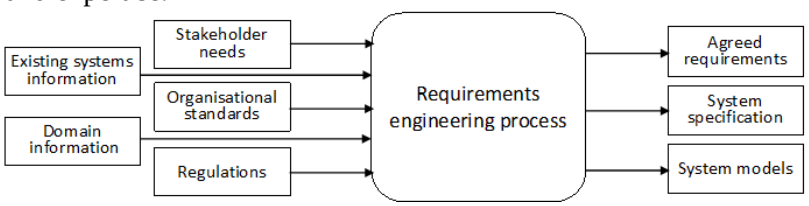


Figure 2.1 - Inputs and outputs of the requirements engineering process, taken from [7]

A process is frequently described using different models and often varies from one organization to another, or sometimes even within the same organization. The technologies and methods used, the culture of the organization, the disciplinary involvement and the type of application domain are the main influence factors that lead organizations to adapt their RE processes to fit their specific needs [8]. Despite the differences, most RE processes can be abstractly described by a general set of main activities that are roughly agreed by the literature [1, 3–5, 7]. In [1], Sommerville proposes the four general activities shown in Figure 2.2.

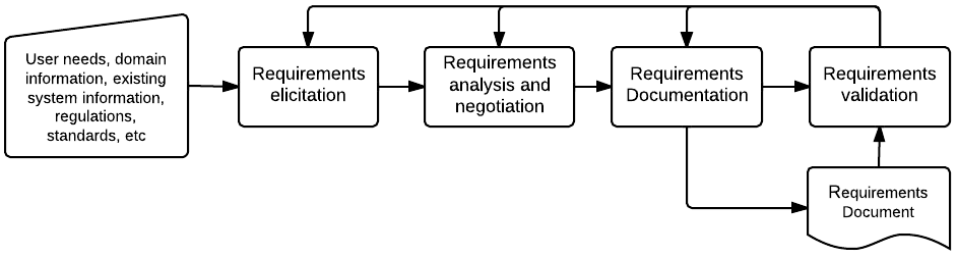


Figure 2.2 - activity model of the requirements engineering process, adapted from [8]

Requirements elicitation: requirements and expectations are discovered by consulting relevant stakeholders, analyzing the documentation and the domain knowledge.

Requirements analysis and negotiation: requirements are analyzed in detail and different stakeholders negotiate to resolve conflicts between viewpoints or other project constraints.

Requirements documentation: the agreed requirements are consolidated and documented. This document should be written in natural language with the support of diagrams, to be understood by a maximum number of stakeholders.

Requirements validation: the documented requirements are analyzed to detect ambiguities or conflicts, ensuring their consistency and completeness.

Sommerville [1], and Nuseibeh and Easterbrook [14] described these core activities as being interleaved, iterative and spanned across the entire software systems development life cycle. To manage the requirements changes and evolution across time, Sommerville identified an additional process of requirements management that should run in parallel with the above [8]. This process includes activities to manage requirements' traceability and analyze the impact and cost of changes.

When designing a new RE process, these high level activities must be considered to make sure that the business needs are explicitly defined, the individual responsibilities are clear, methods and techniques for allowing the identification and communication among all stakeholders are being used, and the requirements changes are properly managed [8].

2.1.4. Requirements Engineering approaches

The RE processes discussed in the previous section may be guided by requirements methods and techniques. These methods are characterized as '*systematic approaches to documenting and analyzing the system requirements*' [8]. They produce system models, which are the main bridge between the analysis and the design stages.

A method is usually associated with a single notation (e.g. controlled natural language [20], Z [21], B language [22], the Unified Modeling Language (UML) [23], i*[24]) that provides a meaning to the expressed requirements. The precision and the understandability of the notation used are two important properties that a method should have to properly model a problem [3]. Similarly to the RE processes, there is no ideal requirements engineering approach [8]. However, it is important to carefully select it, as it will affect (or even restrict) the set of phenomena that can be identified and modeled [14]. A very brief summary of some of the available approaches is presented below.

Viewpoint-oriented requirements engineering is an approach that takes into account different *points of view*, i.e., the different stakeholder's perspectives of the system. Therefore, a viewpoint can be described as a collection of information about the problem, the domain, the system or the environment from a particular perspective. This approach gathers a set of requirements and concerns (or NFRs) from a diversity of sources, which enables the detection of conflicting requirements and reduces the chances of an incomplete set of requirements [8].

Scenario-oriented requirements engineering uses scenarios, which are step-by-step descriptions of stakeholders' interactions with the system, to formulate and complete the system requirements. This approach uses natural language and is generally the most appreciated by users, as it is easy to understand and relates to real-life examples [1].

Aspect-oriented requirements engineering is based on the modularization of crosscutting concerns, known as aspects. Crosscutting concerns are properties, or characteristics, of the system that do not align with the decomposition criteria promoted by classic software development approaches, such as object-orientation, and end up scattered along several different components. Examples of crosscutting concerns might be the usability or the performance of a system. The key advantage of this approach is that it enables these concerns to be understood, reused and modified independently of their components, emphasizing the modularity of the system [25]. Although there is no "de facto" notation for this approach, a few methods exist, like Theme/DOC [26], MATA [27] or AoURN [28].

Object-oriented requirements engineering uses objects to represent the system requirements. An object is an entity defined by a state and a behavior. It has a set of attributes, operations and interfaces to interact with other objects. This approach is generally well accepted by both academia and industry due to the straightforward mapping between real world objects and system entities. The Unified Modeling Language (UML [23]) is the de-facto standard notation used for object-oriented modeling [8].

Goal-oriented requirements engineering uses system goals (be them functional or non-functional properties a system should achieve) to represent the requirements. The key benefits of this approach are the fact that it enables different levels of abstraction of the problem, allows for the completeness of requirements to be measured, and is considered intuitive for stakeholders. There are several notations that can be used, like KAOS [29] and i* [30].

Formal based requirements engineering uses formal methods, based on logic or other mathematical techniques, to specify and validate requirements. This is generally used in the development of critical systems, as it strongly contributes to the accuracy, reliability and robustness of a system [8]. Examples of formal notations are Z [21] or B [22] languages.

2.2. CMMI Overview

The effectiveness level of an organization to develop quality products or services is directly related to the maturity of their processes. In this context, maturity models and standards for Software Process Improvement (SPI) offer companies the possibility to measure and improve their software quality and processes. In [4], Wendler examined maturity models according to several dimensions, and summarized his research as follows: "*Maturity models describe and determine the state of completeness or perfection (maturity) of certain capabilities (...) and define simplified maturity stages or levels which measure the completeness of the analyzed objects via different sets of (multi-dimensional) criteria*".

At the moment, the most popular maturity model is the **Capability Maturity Model Integration** (CMMI) [4]. This reference model is a collection of best practices that can be used to guide process improvement across a project, division, or an entire organization. It was developed by product teams with members from industry, government, and the Software Engineering Institute (SEI) to address some limitations of its ancestor, the CMM model [31], and is currently in version 1.3. The CMMI model actually comprises three reference models, commonly designated by constellations (Figure 2.3):

- **CMMI-DEV** [2] is concerned with the development process of products and services. It addresses the development activities throughout the life cycle, from conception to delivery and maintenance.

- **CMMI-ACQ** [32] manages the acquisition process of products or services. It includes practices and terminology that address supplier sourcing, agreement activities and managing the acquisition of capabilities.

- **CMMI-SVC** [33] is targeted at service providers. It addresses activities to manage internal and external services, like the service delivery, continuity and transition, incident resolution, prevention and addressing capacity and availability management.

Each constellation consists of a collection of process areas specifically chosen to improve a given business need within the CMMI model. A process area specifies goals and practices for improvement, but does not specify how to do it. For this reason, CMMI is mostly a *descriptive* model, rather than a *prescriptive* one. This means that it describes the end result (what to accomplish) but does not prescribe the method to get there. In Figure 2.3, it is also noticeable the existence of three kinds of process areas. Some can be common to all models — *core process area* (e.g. Requirements Management) — others are shared by two of them — *shared process area* (e.g. Supplier Agreement Management) — or even used individually by one model — *specific process area* (e.g. Requirements Development).

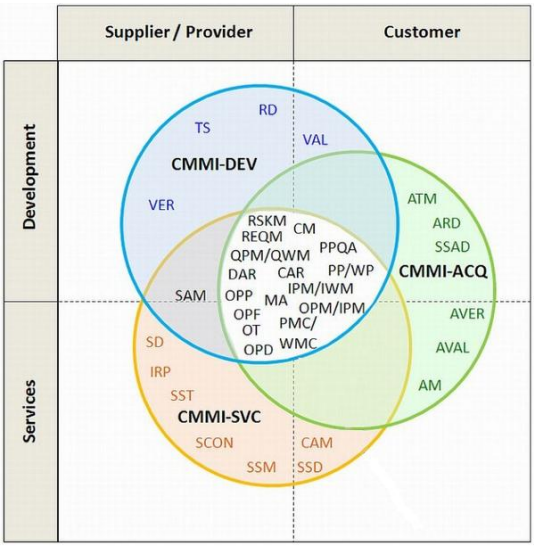


Figure 2.3 - CMMI constellations, adapted from [110]

Each area contains generic and specific goals. The generic goals' practices are expected to be found across any process area. On the other hand, the specific goals have practices that can only be found in that process area. A process area also has informative components, for example sub-practices or typical work products. Such components are shown in Figure 2.4.

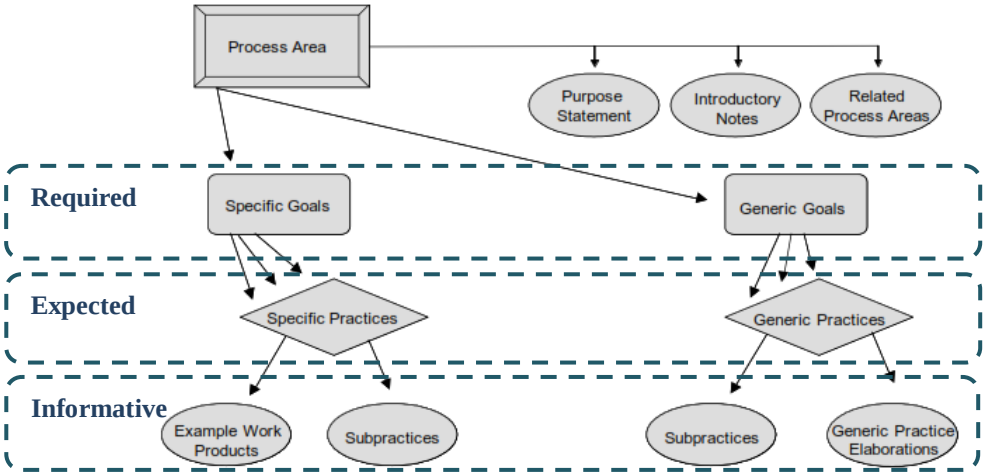


Figure 2.4 - CMMI model components, adapted from [2]

It is important to note that a process area is not a process, nor its practices are a process description. Instead, they should be regarded as characteristics of effective processes. The actual processes used in an organization depend on factors such as its size, structure, business goals and application domain. In particular, they do not map one to one with process areas [2]. A process can satisfy one or more practices from distinct process areas.

2.2.1. CMMI Levels

Under the CMMI methodology, processes are organized according to levels, describing an evolutionary path recommended for an organization. Each level can be awarded to a process area or to an organization, depending on the representation model chosen.

The **continuous representation model** is characterized by ***capability levels (CL)*** granted to process areas, which enable the organization to focus its improvement process area by process area, from capability level 0 (incomplete) to capability level 3 (defined) [34]. However, there are some limitations when selecting process areas due to the dependencies among them.

On the other hand, the **staged representation model** takes the company through an ordered sequence of ***maturity levels (ML)***, ranging from level 1 to level 5, which characterize the organization's behavior. To reach a certain maturity level, a company must successfully achieve the goals and practices of the set of process areas associated to that level. This systematic structured approach ensures that a foundation had been laid for the next stage and, for this reason, it is usually the chosen representation when implementing CMMI [34].

Table 2.1 - CMMI Levels, taken from [2]

Level	*Capability Levels Continuous / Process*	*Maturity Levels Staged / Organization*
Level 0	Incomplete	
Level 1	Performed	Initial
Level 2	Managed	Managed
Level 3	Defined	Defined
Level 4		Quantitatively Managed
Level 5		Optimized

At **maturity level 1 (ML1)**, the organization is greatly dependent on the abilities of its employees. Its processes are ad hoc and chaotic and usually abandoned in time of crisis. Although the organization manages to deliver its products, they are usually over schedule or budget.

Maturity level 2 (ML2) is characterized by processes for the project, which is performed and managed according to its documented plans. Commitments are obtained from relevant stakeholders and the project's progress is visible to management at defined points.

Reaching **maturity level (ML3)** means that the processes are rigorously defined and normalized for the whole organization using standards, procedures, tools, and methods. Each project is adapted to this organizational set of processes using tailoring guidelines.

At **maturity level 4 (ML4)** the quality and the processes' performance is quantitatively measured and controlled using selected measures that are collected and statistically analyzed in order to ensure the project's predictability.

Finally, **maturity level 5 (ML5)** is the last level an organization can reach and is focused on the continuous improvement of the organization's processes based on the quantitative understanding of the organization's performance.

These organizational maturity levels are dependent on the capability levels of their processes. For instance, to reach maturity level 2 all the processes are required to perform at capability level 2, whereas for upper maturity levels the capability level 3 is required for the target processes. This interrelationship can be consulted in Appendix A.

Each representation model has its own advantages [35]. On one hand, the staged representation can be used for marketing purposes, since it offers a classification level easy for advertising and serves as a basis for comparison with other organizations. On the other hand, the continuous

representation provides flexibility, since it allows the company to select the area(s) and order of implementation that best meets the business objectives, improving different areas at different rates. Nevertheless, even if the company chooses the staged representation, it can address other process areas independently of which maturity level they belong.

2.2.2. SCAMPI Appraisals

The method used to assess an organization's processes is usually referred to as **SCAMPI** (Standard CMMI Appraisal Method for Process Improvement). An appraisal is an activity that helps identifying strengths and weaknesses of processes by examining how closely they relate to CMMI best practices [36]. They are a catalyst for improvement, as they help to recognize future process improvement opportunities. Organizations are often submitted to these appraisals to measure the progress of their improvements or to be awarded a maturity level. SCAMPI assessments must be conducted by a certified *Lead Appraiser* and were design to be repeatable, to ensure the consistency of results, and can be classified according to three kinds:

- **SCAMPI A** appraisals focus on the "institutionalization" of the processes, required for awarding a maturity or a capability level [36]. This method uses rigorous standards for detailed data collection and the usual outcomes are a documented set of strengths, weaknesses and key issues found for future improvement.
- **SCAMPI B** focuses on the "deployment" of the processes. It is used to assess the progress towards a targeted level at a lower cost and rigor than SCAMPI A. It uses relaxed standards for data collection, producing a set of strengths, weaknesses and issues found, as well as an indication of the likelihood that the company would pass in a SCAMPI A appraisal [37].
- **SCAMPI C** is the less rigorous appraisal and is focused on "approaching" the processes. It is shorter, more flexible and usually conducted for a quick gap analysis or to monitor the implementation of a new process. The outcomes are generally related to the strengths, and weaknesses found on the processes, as well as improvement actions recommended [37].

These three approaches can be complementary for a progressive implementation of CMMI. For example, "*starting with a SCAMPI C reviewing the process descriptions, then a SCAMPI B investigating their deployment to projects, finally leading to a formal benchmarking event focused on institutionalization of the practices across the organization*" [37].

2.2.3. Performance Results

Process improvement through CMMI has proven impacts on the organizations. It can be applied in small or large companies of a variety of industries, and is compatible with other technologies, such as Agile methodologies and ISO Standards. In 2006, the Software Engineering Institute (SEI) collected data from 35 organizations using CMMI-based process improvement, most of them with high maturity, and summarized the performance results in Table 2.2. These results evidence that it benefits organizations in many aspects, particularly in productivity.

Table 2.2 - Performance Improvements over Time by Category, taken from [7]

Performance Category	Median Improvement
Cost	34%
Schedule	50%
Productivity	61%
Quality	48%
Customer Satisfaction	14%
Return on Investment	4:1

According to the latest Maturity Profile Report [38], with data gathered between 2007-2013, about 7800 SCAMPI appraisals were conducted in 88 countries, with particular predominance in China and in United States. In 2012 a steady rise was noticeable, reaching its peak with 1412 SCAMPI A appraisals. Most of the organizations evaluated were commercial, and more than a

half of them reached maturity level 3 (defined). Regarding the size of the organization, most appraisal reports came from small to medium scale organizations, with less than 100 employees.

2.2.4. Problems and Limitations

Staples *et al*. conducted an exploratory study on why organizations do not adopt CMMI [39]. The results indicate that the high cost and time needed to implement this model are the main reasons mentioned, especially by small scale organizations. This is something that contrasts with the statistics in the latest CMMI maturity report [38], as they revealed most appraisals were conducted on companies with less than 100 members. The overhead in documentation and bureaucracy is also an obstacle to the adoption of this model, as it increases the number of outcomes that need to be produced during the development of a project.

During the discussion panel of the III CMMI Portugal conference [40], the main challenges and the future of CMMI were debated. One of the main concerns was whether this model is going to become more commercial than a synonym of excellence. It is known that some organizations don't understand the goal of CMMI, and take this model as quality stamp rather than a process improvement tool. Moreover, following CMMI like a recipe by producing a lot of documentation and collecting a lot of data does not lead to improvements. Organizations need to tailor their processes and measures according to their business goals in order to really improve.

The overcome of these challenges relies on more accurate evaluations that are based on evidences and especially on the rigor of evaluators when assessing if real improvements have been achieved. It is not sufficient to check for the compliance of the processes with the recommended practices. Furthermore, the involvement of the community on improving the model is also valuable to overcome existing limitations on the current version. According to [41], both the model and the appraisal methods are evolving to a higher maturity state, taking us one step further towards version 2 of CMMI.

2.3. Requirements Engineering in CMMI

Requirements Engineering has been acknowledged as a foundation for Software Quality [42] and so, many maturity models also try to address this issue. As previously mentioned, CMMI establishes two process areas that target RE process improvement: Requirements Management (REQM) and Requirements Development (RD). These process areas do not require any specific development life cycle model and are closely related to other core process areas. In this section these process areas will be described and reported problems and limitations briefly explained.

2.3.1. Requirements Management and Development Process Areas

Requirements Management (REQM) is a core process area required at maturity level 2, and its main goal is to manage all the requirements of the product (or service) and its components [2]. This goal is accomplished with 5 specific practices that aim at understanding and obtaining commitment to the requirements, and to ensure the alignment of the project plans and final product with these requirements. Additionally, bidirectional traceability between requirements should be maintained and, as requirements evolve, their changes should be managed with change requests that are documented along with their rationale. The specific goal and practices required by this process area are shown in Table 2.3.

Table 2.3 - Goal and practices of Requirements Management process area

Goals	*Practices*
SG 1 Manage Requirements	SP 1.1 Understand Requirements
	SP 1.2 Obtain Commitment to Requirements
	SP 1.3 Manage Requirements Changes
	SP 1.4 Maintain Bidirectional Traceability to Requirements
	SP 1.5 Ensure Alignment between Project Work and Requirements

A defined process to manage requirements should perform at least at capability level 2. This means that it should be planned and executed in accordance with policy by skilled people who have adequate resources to produce controlled outputs [2]. Plus, it should involve stakeholders and be reviewed to evaluate the adherence to its description.

Requirements Development (RD) is a specific process area required at maturity level 3. The purpose of this area is to elicit, analyze and specify customer, product and product component requirements for the entire life cycle of the project [2]. This is accomplished with 3 specific goals and 10 expected practices (see Table 2.4). The first goal aims at eliciting the needs, expectations and constraints of all stakeholders so that they can be consolidated into a documented set of customer requirements. The second goal refines the customer requirements into product and product component requirements, including its interfaces. The last goal supports the other two by setting practices to analyze and validate all the requirements, in accordance with the user's intent and environment. This area gives special attention to quality attributes that can influence the architecture, as well as on the risks related to requirements.

Table 2.4 - Goals and practices of Requirements Development process area

Goals	*Practices*
SG1 Develop Customer Requirements	SP 1.1 Elicit Needs
	SP 1.2 Transform Stakeholder Needs into Customer Requirements
SG2 Develop Product Requirements	SP 2.1 Establish Product and Product Component Requirements
	SP 2.2 Allocate Product Component Requirements
	SP 2.3 Identify Interface Requirements
SG3 Analyze and Validate Requirements	SP 3.1 Establish Operational Concepts and Scenarios
	SP 3.2 Establish a Definition of Required Functionality and Quality Attributes
	SP 3.3 Analyze Requirements
	SP 3.4 Analyze Requirements to Achieve Balance
	SP 3.5 Validate Requirements

A process to develop requirements should perform at capability level 3. This means that it is a managed process tailored from the organization's set of processes according to specific tailoring guidelines [2]. It differs from a capability level 2 process in rigor and scope. While at capability level 2 the procedures used might be very different in each instance of the process (e.g., on a particular project), at capability level 3 the procedures are chosen from the organizational processes according to tailoring guidelines (e.g., on project type). Therefore, a capability level 3 process is more consistent and should clearly state its purpose, inputs, entry criteria, activities, roles, measures, verification steps, outputs, and exit criteria.

Even though Requirements Management and Requirements Development are two distinct process areas implemented at two different maturity levels, they are strongly related and can easily be mixed. Requirements Management is recommended at a lower level because keeping track of the requirements and its changes throughout the project life cycle is considered a top priority. Practice SP1.2 of Project Planning process area states that "*The estimates should be consistent with project requirements to determine the project's effort, cost, and schedule*". Therefore, the management of requirements is indispensable for improving Estimation [43].

'*REQM takes care of managing the requirements developed in the RD stage and subsequent stages of the life cycle by handling requirements changes and sign-offs in an organized manner. As and when RD changes the requirements, RM manages and controls the requirement changes and assesses its impact on other work products and the phases of the life cycle*' [44]. The tools used for support also contrast. A REQM process uses tools that track requirements and its changes, while RD needs tools for specifying them, such as modeling and prototyping tools. These process areas distinct characteristics are summarized in Table 2.5.

Table 2.5 - Main differences between REQM and RD process areas

	REQM	*RD*
Purpose	"**Manage**" the requirements of the project's products and product components. **Identify** inconsistencies between requirements and project's plans/work products.	"**Produce**" customer, product, and product component requirements. **Analyze** the customer, product and product component requirements.
Goals	Manage Requirements.	Develop Customer Requirements, Develop Product Requirements, Analyze and Validate Requirements.
Tools	Requirements **tracking** tools, traceability tools and bi-directional matrix.	Requirements **specification** tools, simulators, modeling/ prototyping tools, scenario definition tools.
Outcomes	Requirements, requirements traceability matrix, change requests, formal commitment to requirements.	Customer, product, product-component and Interface requirements, functional architecture, quality attributes.

The RE concepts of Section 2.1 can be found analyzing the practices of these process areas. Regarding the terminology used, Sommerville's classification of requirements as *User* and *System* requirements can be matched to the concepts of *Customer* and *Product* requirements, which are clearly separated in two distinct goals of the RD process area. Moreover, the distinction between *Functional* and *Non-functional* requirements is implicit in practice SP 3.2 with the identification of *quality attributes*, a type of non-functional requirement that will influence the architecture of the system.

The main process activities identified in Section 2.1.3 are also recognized throughout these process areas. The RD process area specifies practices for the elicitation, analysis, specification and validation activities. The management and documentation of requirements is described by the REQM process area. This separation into two individual process areas is similar to the general RE process proposed by Somerville [8] that detaches the development from the management of the requirements.

Being a "descriptive" and not a "prescriptive" model, these CMMI process areas don't require any particular RE technique to be employed. The organization's practitioners are free to choose the methods used to address the practices recommended.

2.3.2. Problems and Limitations

Even though these RE basic concepts are present in CMMI, some authors believe that the model has room for improvement [43], [45].

Linscomb questions whether CMMI defines RE maturity progression the best way [45]. Although he acknowledges that the RE related process areas are properly placed under maturity levels 2 and 3, he considers that the order and the separation between management and development is not correct, as it does not comply with the literature. According to the author, managing requirements at ML2 requires certain practices of ML3 (like elicitation and analysis) to be institutionalized first, so that requirements are mature enough to be managed. In fact, organizations at ML2 do perform these ML3 activities, even if not systematically.

To bridge these gaps, Linscomb recommends a review of the model classification of RE maturity by redefining the related process areas as "Basic RE" and "Advanced RE", respectively place at ML 2 and 3. The Basic RE would consider activities like eliciting, analyzing, documenting and getting approval of requirements from appropriate stakeholders. This process are would also recommend managing its changes and high level traceability. The Advanced RE would concern more sophisticated RE practices, like: establishing requirements elicitation techniques according to project profiles; providing trained staff on requirements;

establishing low-level traceability of requirements from all development life-cycle phases and other RD practices already proposed by the model.

Also Buglione *et al*. [43] identified a set of possible improvements to the CMMI Requirements Management and Development process areas, based on other RE maturity models. They claim that an inner limitation of any model is its scope and approach for describing a certain issue. Therefore, they proposed integrating CMMI RE process areas with specific RE maturity models, using their *Living EnGineering prOcess* (LEGO) approach.

The main improvements identified through LEGO are mostly associated with the RD process area, rather than the RM, and concern practices or subpractices for: Stakeholders Identification and engagement; Definition of requirement attributes (e.g. priority, risk, status); More specific requirements classifications (e.g. functional, quality, technical); Identification of volatile requirements; Definition of a (standard) document structure; Establishing criteria for writing better requirements; Documenting technical and organizational attributes specific to a Project; Suggesting the use of workflow environments for sharing information on requirements.

To improve the model, the authors integrated these suggestions in the current RD goals and practices, proposing a new enhanced process area. However, these new methodology was never applied in real projects.

2.4. Summary

Throughout this chapter, the basic concepts of this dissertation were introduced. RE was described as the branch of software engineering concerned with the elicitation, analysis, specification, validation and management of software requirements. Since this inital phase of software development is aknowledge as a foundation for project success, also the CMMI maturity model attemps to address it through the definition of two process areas: the Requirements Development and Requirements Management.

Even if the impact of this model on organizations seems to be substancially beneficial in software quality, some companies chose to not addopt it because they considered that the model is too costly (in terms of money, effort and time need) to be implemented by small organizations.

Moreover, some authors believe that there is still room for improvements in the model where RE is concerned. In particular, it has been questioned whether the CMMI properly represents RE maturity progression and if the practices of other specific RE maturity models could be used to enhance the current process areas.

3. Requirements Management and Development in Portugal

To get an overview about the interest in the CMMI model in Portugal, the process maturity profile reports published by the CMMI Institute were analyzed regarding the number of appraisals performed in Portugal and reported to SEI. According to these reports, the Portuguese interest in process improvement through the CMMI framework started in 2005 and has nearly doubled in the last 3 years (see Figure 3.1). The latest maturity profile report [38] shows that there were 31 appraisals conducted in Portugal mostly assessed with maturity level 2 or 3, except for 4 organizations that reached the highest maturity level, as shown in Figure 3.2.

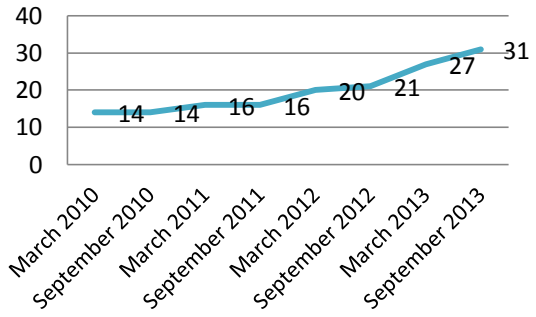


Figure 3.1 - Number of Appraisals performed in Portugal and Reported to SEI

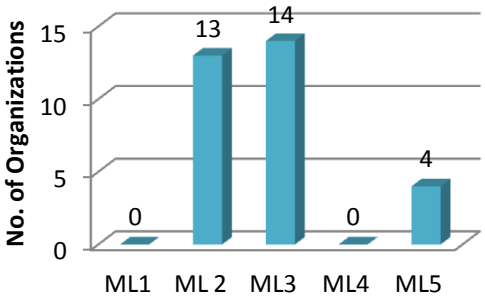


Figure 3.2 - Maturity Profiles by reporting organizations in Portugal

These numbers reveal that the amount of Portuguese companies appraised is rather small and the interest in CMMI maturity has been steadily rising.

In this chapter, an empirical study on CMMI and the RE techniques in Portugal is provided in Section 3.1, along with a brief discussion of the main results found in Section 3.2.

3.1 Empirical Study on CMMI and the RE techniques

To further investigate the state of practice in Requirements Development methodologies compliant with CMMI in Portugal, we conducted a survey to evaluate the current industrial trends in Requirements Engineering techniques and its relationship with CMMI maturity. This section describes this study, the general findings and the possible threats to validity.

3.1.1 Instrument Design

The survey was conducted during the III CMMI Portugal conference held in Lisbon in November 2013[1]. This conference was organized by the SPIN Portugal group with the purpose of sharing knowledge on process improvement and CMMI through informal scientific meetings.

During this conference a questionnaire was made available to all the participants in both online (Google Forms) and paper versions to promote the number of respondents. Although the participants had varying degrees of experience, we assumed that all of them shared a common knowledge on process improvement through CMMI and were aware of their company's processes, given that they were attending a conference dedicated to this subject.

The survey (in Appendix B) consisted of both open-ended and close-ended questions. The close-ended questions provided multiple choice answers to gather the techniques and tools used when eliciting, modeling and validating customer, product and non-functional requirements. To collect additional data, the option "Other" was also made available in each question. The aim of the open questions was to gather the main problems when developing requirements and possible suggestions or additional comments on the definition of a requirements development process. For each question, the ambiguous terminology was explained to avoid misunderstandings. To

[1] https://sites.google.com/site/conferenciascmmiportugal/

conclude, the results of the survey were made available to participants, provided that the email address was given.

3.1.2 Data Collection and Analysis

An average of 45 participants attended the III CMMI Portugal conference, and all received both the survey URL and the paper versions. Two additional CMMI certified companies were directly contacted via email to increase the number of responses and get more accurate results. A total of 20 responses from 15 different companies was gathered and analyzed. Only 6 of these 15 companies revealed a CMMI level certification, while the remaining were not certified. One participant didn't indicate his affiliation and therefore was excluded from the study. The responses were stored in the database of the online survey tool used and later interpreted with the support of a statistical analysis tool (SPSS).

Both qualitative and quantitative analysis was conducted on the data gathered. Data was analyzed to ascertain the percentage of companies using each technique or tool, through a qualitative comparison of CMMI certified and non-certified organizations. In addition, the open-ended responses reporting the main problems and suggestions were analyzed qualitatively due to its nature.

3.1.3 Threats to Validity

The overall findings and conclusions of this study may be threatened by a number of factors. First, the small number of responses gathered makes the sample not representative of the population as a whole. In particular, only 6 out of the 31 exiting CMMI certified companies in Portugal answered the survey. Moreover, the survey was conducted at the III CMMI Portugal conference, which limits the variety of subjects on companies that are already interested in process improvement through CMMI. Besides, it was not asked if the companies were certified with another model. The background or roles of the participants were not questioned and may not be directly related to requirements engineering.

To mitigate this threat, we have included the definitions of the main concepts used and made sure the questions were comprehensible by proof reading the survey with external reviewers, before being accepted for distribution. Finally, we intend to mitigate the threats by validating the conclusions via expert feedback on Requirements Engineering. However, one should be aware that these conclusions represent current industrial trends and, therefore, may not be generalized across time.

3.1.4 Results

The results will be presented according to the questions of the survey.

1. Which techniques are used in your organization to identify the stakeholder needs?
Considering the elicitation techniques, the usage of interviews and document analysis are the most common techniques between both kinds of organizations. Prototypes are also fairly accepted. It is interesting to note that the elicitation of requirements using workshops, design thinking, scenarios and storyboards is more popular for organizations CMMI certified.

2. Which techniques are used in your organization to model the customer requirements?
Regarding the modeling techniques for customer requirements, the analysis evidence that modeling customer requirements using the natural language seems to be the top choice among organizations. The usage of UML is also well accepted to support the natural language. In addition, User Stories were a technique only reported by certified organizations. Inversely, goal-oriented techniques were barely mentioned and seem to be used only by non-certified companies.

3. Which techniques are used in your organization to model the product requirements?
Analyzing the answers regarding product requirements modeling, we found almost all of the respondents from all organizations indicated that they modeled product requirements using

natural language and Use Cases/Scenarios. Additionally, Object-Oriented Approaches (like UML), also seem to be well accepted by both certified and non-certified companies. In contrast, viewpoint-oriented, goal-oriented techniques or formal methods were hardly ever mentioned.

4. *Which requirements validation methods are used in your organization?*

Considering the requirements validation techniques, the numbers indicate that most certified organizations validate their requirements using a combination of techniques, particularly using requirements testing and Scenarios or User Stories. It is also interesting to note that formal validation techniques, like inspections and model-based V&V, are more popular among certified organizations.

5. *Which techniques are used in your organization to develop Non-Functional Requirements (NFRs)?*

Analyzing the answers regarding Non-Functional Requirements modeling, it is clear that the usage of natural language is the preferred technique. Moreover, goal-driven use cases were also a technique indicted by 33% of the respondents from certified organizations. Nevertheless, some certified and non-certified organizations stated that still don't address this kind of requirements.

6. *Which tools does your organization use for requirements development?*

Concerning the tools used to support the modeling activities, MS Office was the most reported tool by both certified and non-certified organizations. Case Tools to specify and model requirements also seem to be well-accepted by both organizations. Conversely, requirements management tools, like RedMine, ReqPro, JIRA or TestLink, were mentioned mostly by certified organizations.

3.2. Discussion

Although the number of companies certified with a CMMI level in Portugal is rather small, the interest in this maturity model has been rising, doubling the number of appraisals during the last 3 years. To further investigate the state of practice in Portuguese companies, we conducted a survey to evaluate the current industrial trends in Requirements Engineering techniques and its relationship with CMMI maturity.

The main findings of this study show that the differences in RE techniques used by CMMI certified and non-certified organizations is subtle. Regarding the elicitation methods, interviews and document analysis seem to be the most popular, despite of the presence of a CMMI certification. However, scenarios and storyboards are more used by the certified companies. The numbers also indicate that the validation of requirements is more systematically conducted among certified companies, given that most of them combines several techniques like requirements testing, prototyping and internal inspections. It is also interesting to note that goal-oriented techniques are not yet fully accepted by the industry, with an exception on non-functional requirements, which are modeled using goal-driven use cases by 33% of the certified companies. Still, the use of natural language seems to be the top choice for modeling every kind of requirements, most likely because of the ease of understanding for both the customer and the development team. Nevertheless, UML-based techniques and its tools are generally used for support, regardless of the existence of a CMMI certification.

Finally, tools for requirements management, like TestLink, ReqPro, JIRA and Redmine, were more mentioned by CMMI certified organizations. This might be due to the recommend practices of the Requirements Management process area, which is required at a low level maturity.

4. Requirements Management and Development at Altran

Altran Portugal provides consulting services in several business sectors such as financial, telecommunications & media, public administration, industry and utilities. The company offers support from planning to manufacturing and its projects are focused on four key business areas: Intelligent Systems, Information Systems, Lifecycle Experience and Mechanical Engineering.

Concerned with improving the quality and value of the services and products provided, Altran Portugal has started its process improvement strategy in 2002 with the implementation of its Quality Management System that granted the company with ISO 9001 certification. In 2003, they developed a Project Management Methodology based on PMI (PMBoK) [46], which was later upgraded to be aligned with the CMMI-DEV 1.3 reference model [2].

Throughout this Chapter, the implementation of CMMI at Altran is briefly outlined in Section 4.1, along with a detailed description of the current Requirements Management process, in Section 4.2. The relationship of this process with other existing processes is analyzed in Section 4.3. A CMMI based assessment of the presented process and its derived improvement plans are provided in Sections 4.4 and 4.5, respectively. To conclude, Section 4.6 summarizes this Chapter.

4.1. CMMI at Altran

The implementation of CMMI-DEV at Altran Portugal opened up the company to new business opportunities and improved the quality, budget and schedule of the projects specifically in the areas of software and systems engineering. In 2012, Altran Portugal was first submitted to a SCAMPI C appraisal, which was followed by a SCAMPI A that rated the Solution Center department with CMMI-DEV 1.3 maturity level 2.

The CMMI model was applied by Altran through the creation of the Delivery Management System (SGD[2]). The SGD maintains Altran's improvement strategy and promotes a gradual but consolidated growth in business. It comprises a collection of CMMI-compliant processes with activities, inputs, outputs, responsibilities, rules and other process elements that guide the development and maintenance projects of the Solution Center.

4.2. The Requirements Management process

The Requirements Management process [47] is one of the processes integrated in the SGD that complies with the rules of the Requirements Management process area required to achieve CMMI maturity level 2. This process will be described in this section along with the roles and tools associated to each task. The detailed description of the process will be followed by a Gap Analysis to identify the level of compliance to CMMI REQM and RD process areas.

4.2.1. Roles and Responsibilities

The main roles throughout the development life cycle of a project at Altran and its associated responsibilities are listed and summarized next. Figure 4.1 illustrates the main interaction between the identified roles.

- *Business Manager (BM):* Acts as the sales representative. He is responsible for detecting the business opportunity and handovers the customer requirements to the Practice Manager. This handover is typically done through minutes of meetings and customer's documentation. The business managers and the practice managers are responsible for producing an offer/proposal that is delivered to the customer.

[2] Sistema de Gestão de *Delivery*

- *Solution Center Director or Delivery Manager:* Acts as the responsible for the solution center area. He is responsible for assigning and informing the Project Manager about the project details.
- *Practice Manager:* Acts as the pre-sales representative. He is responsible for the solution that is thought to meet the customer requirements as stated by both the customer and the Business Manager. This solution is presented in the offer/proposal that is delivered to the customer. The Practice Manager is also responsible to handover the requirements to the Project Manager.
- *Project Manager (PM):* Responsible for the creation of the requirements list and the preparation, update and presentation of the Project Management Plan. The Project Manager is also responsible to support and train the team on the process, as well as promoting meetings to get stakeholder's commitment and approval.
- *Development Team:* Responsible for the development of the solution, the creation and analysis of change requests, and the maintenance of the requirements traceability.
- *Test Team:* Responsible for developing the test plan, maintaining the bidirectional traceability of the requirements, and performing and reporting the tests to assure that no inconsistencies exist between the work products and requirements.
- *Customer:* Responsible for the approval of the requirements list, the requirement and test specification, the work products, create and approve change requests and accept the project and final deliverables.

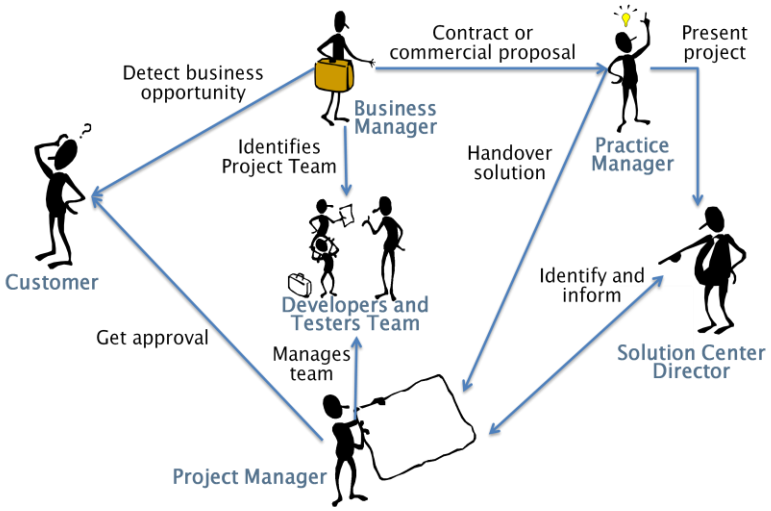


Figure 4.1 - Interaction among roles in a development project at Altran Portugal

4.2.2. Activity Workflow

The Requirements Management process starts with the detection of a business opportunity by the Business Manager, who elaborates a business proposal (or the request for proposal) and writes down the meeting minutes with the customer. These documents are the input for the first activity. The REQM process workflow is illustrated in Figure 4.2. This Section describes each activity systematically, with a description, inputs, outputs and a table listing the involved roles and their responsibilities.

Customer requirements

Description: During the elaboration of the proposal, the BM and the Practice Manager identify the high level requirements that meet both the customers' needs and the solution proposed. Then, they present the project to the Delivery Manager, who will decide for its approval and assign its PM and team. The BM and Practice Manager shall brief the PM with the details.

Inputs: The proposal, meeting minutes or other sources (formal or informal) of information.

Outputs: The high level customer requirements and inputs for the project charter.

Table 4.1 - Roles and Responsibilities of Customer Requirements Activity, taken from [41]

Role	*Responsibility*
Delivery Manager	Assign and inform PM
Practice Manager	Handover of information to PM
Business Manager	Handover of information to PM
Project Manager	Accept assignment, and understand project requirements

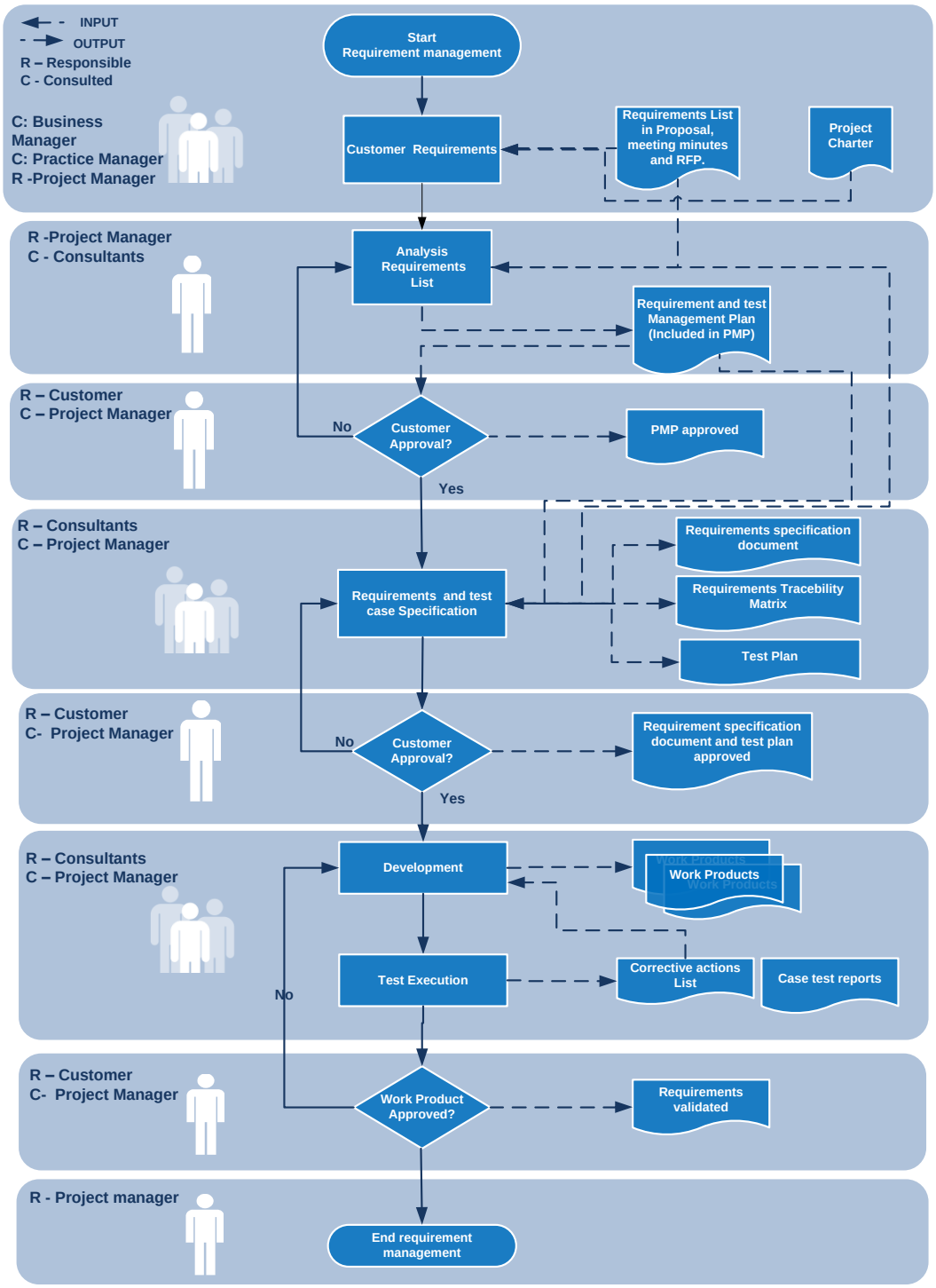


Figure 4.2 - Requirements Management Workflow, taken from [47]

Analyze Requirements list

Description: After the PM receiving the project, there are internal meetings to create the requirements list and present the proposal to the team. This list is the result of the breakdown of customer requirements into work packages and work packages into deliverables. In this meeting the granularity of the requirements is specified, the alignment between the requirements and the plans is ensured and the PMP baseline will be filled. The PMP includes test phases, mechanisms that allow the customer to validate requirements and how they deal with changes and defects.

Inputs: The project charter, the customer requirements, project constraints, assumptions and other related documents.

Outputs: The requirement list, Meeting Minutes, mechanisms for validation of requirements.

Table 4.2 - Roles and Responsibilities of Analyze Requirements List Activity, taken from [47]

Role	*Responsibility*
Project Manager	Prepare the information about the requirements to include in the project plan Responsible for conducting an internal meeting to present the project plan Obtain the commitment from the project team for the PMP
Project Team	Attend the PMP review meeting and commit to the PMP

Approval of initial requirements list

Description: The PM books a kick off meeting with the customer to present the project management plan and its initial requirements list, and obtain his formal approval.

Inputs: Project Kick off presentation, the PMP baseline.

Outputs: Customer approved PMP.

Table 4.3 - Roles and Responsibilities of Approval of initial Requirements list Activity, taken from [47]

Role	*Responsibility*
Project Manager	Present the PMP and get the customer approval
Customer	Review and approve the PMP

Requirements and test case specification

Description: Transform the requirements list into specifications by refining the requirements and document them formally in the Requirements Specification Document. This document might be updated through the following phases of the project and is essential to obtain a common understanding of the final product by all stakeholders, as it bridges the gap between customer's needs and the constraints imposed by the technology, design and development team. Any inconsistencies and unclear issues should be solved during this phase.

Sub-activity: Capture Customer Requirements
This first step will draw the needs, expectations and restrictions stated by the customer. The team will analyze the requirements list and extract requirements to be developed and formalized. Additional meetings with the customer might be set up if needed. This sub-activity might be unnecessary when the requirements are already specified, for instance by the customer or in a Proposal, being only necessary to verify if they fit the scope, are consistent and well defined.

Sub-activity: Requirements Analysis
The collected requirements are analyzed by the team and translated into product requirements (also known as technical requirements). This includes the analysis of functions, features and restrictions of the product, addressing design decisions constraints, technical difficulties and possible conflicts among requirements or stakeholder's interests. The team is free to choose the techniques used for this analysis, and its results may involve the integration of new requirements, refinement of existing ones or the inclusion of new dependencies.

In the Requirements Specification Document, the requirements should be independent of the designed solution (focused on *what* not on *how*) and must be specified using natural language,

ensuring they are correct, complete, unambiguous, consistent and verifiable. According to Altran, well specified requirements includes: unique ID; name; nature (Functional, Non-Functional); description; dependencies; estimation of effort, pre/post conditions; status (draft, proposed, approved, rejected); history

Afterwards, the set of requirements must be validated by the team to ensure consistency and completeness. Once the PM approves the requirements specification document and assures they are aligned with the PMP, it can be formally approved by the customer.

Sub-activity: Create Requirements Traceability
Traceability is a set of cross-references that relate requirements of different levels (from customer needs to technical requirements specifications, and their associated test cases). It addresses not only the requirements, but also the test cases to indentify the ones that need to be performed to validate a given requirement and, whenever possible, the source code to quickly identify which products must be amended after a change in a requirement, and vice-versa. Thus, it must be bi-directional and it's a vital mechanism to accommodate changes, studying their impact on the project and assure an easier maintenance of the final product. This traceability references can be done automatically using the TestLink and SVN tools.

Sub-activity: Create Test Plan
After receiving the approved PMP, the test team will create the test plan and define the acceptance criteria for requirements (the set of tests to be carried out). Each requirement is associated to one test case that needs to be registered in TestLink, specifying its priority, expected result and associated requirement.

Inputs: Requirements List in proposal, meeting minutes, RFP, Project Charter, PMP.

Outputs: Requirements specification document, traceability matrix and test plan.

Table 4.4 - Roles and Responsibilities of Requirements and test case specification Activity, taken from [47]

Role	*Responsibility*
Test team	Create the test plan (in Testlink) Create the traceability of requirements (in Testlink)
Functional Consultants	Create the requirement specification document (export from Testlink) Create the traceability of requirements (in Testlink)
Project Manager	Validate the requirements specification document Validate the test plan Validate the traceability of requirements

Approval of requirements and test specification

Description: The PM promotes a meeting with the customer to present the documentation and get his formal approval for the requirements specification and test plan.

Inputs: Requirements specification and Test plan.

Outputs: Requirements Specification and Test plan acceptance term signed by the customer.

Table 4.5 - Roles and Responsibilities of Approval of requirements specification Activity, taken from [47]

Role	*Responsibility*
Project Manager	Present/deliver the information. Promote and collect approval
Customer	Approve the requirements specification and test plan

Development

Description: The SW development consultants develop the work products that meet the requirements specification, ensuring its traceability by the use of Testlink and subversion tools.

Inputs: Requirements specification and traceability.

Outputs: Work products, and updated traceability.

Table 4.6 - Roles and Responsibilities of Development Activity, taken from [47]

Role	*Responsibility*
Project Manager	Ensure the conformity of work performed according to plan
Development Team	Develop the work products or solution Maintain the documentation/tools updated Develop the approved project work products changes

Test Execution

Description: The test cases are executed to validate the solution developed and to ensure that it complies with all the requirements. The evolution of tests is tracked on the TestLink. If an anomaly is detected, a defect is generated and analyzed by the development team to uncover its reasons, severity and implications on the project plan. If that defect is solvable without affecting the requirements or the PMP, the team should solve it and register the solution on the system using the tools Mantis or JIRA. Otherwise, if it affects the requirements or the PMP, the PM should analyze its implications and if needed deploy a change request. These defects can be associated to errors detected when running a test case and the changes might be related to the modification of the specification of a requirement, to an incomplete set of requirements or changes in functionality requested by the customer. Further details on defect management are out of the scope of the REQM process.

Inputs: Test plan, test specification, requirements and acceptance criteria and work products or solution to be tested.

Outputs: Report of the tests executed, defects and change requests.

Table 4.7 - Roles and Responsibilities of Test Execution Activity, taken from [47]

Role	*Responsibility*
Project Manager	Monitoring the quality of the test and evaluate test results
Test Team	Executing and reporting the Test on the Testlink Solution Register and follow-up the defects (Mantis or JIRA)

Approval of work products or solution

Description: The PM books a meeting with the customer to present the final work products and get his formal acceptance on the solution delivered. This activity can be performed several times to approve different parts of the project products and the approval can be made even if the product has some defects, depending of what was previously established between the parts. If the customer does not agree with the results a change request may be deployed.

Inputs: Work product and test reports.

Outputs: Acceptance document signed by the customer (might be by email).

Table 4.8 - Roles and Responsibilities of Approval of work products or solution Activity, taken from [47]

Role	*Responsibility*
Project Manager	Present a resume of the tests and their results (for work products)
Customer	Validate and accept the work products or solution

4.2.3. Tools

This Section lists the standard tools typically used by Altran Portugal for its software projects. The user privileges granted to each application are defined according to each project and role.

- TESTLINK – Open source tool used for requirements and test management. All requirements are registered in this tool and organized into directory trees. Test cases are designed and executed against each requirement. The requirements traceability matrix and a requirement specification report can be exported with this tool.
- SVN – Open source tool used for version control. All configuration items (source code) except documentation must be stored and controlled using this application.

- KNOWLEDGE TREE – Document management tool used to store and control all the documents related to a project.
- MANTIS OR JIRA – Change Control/Defect Management tools used to record and control all changes, defects, problems and enhancements of a project.

4.3. Context of Requirements Management process in the SGD

The Requirements Management process is closely related and can be framed in the context of other two processes included the SGD. In this section, these two processes will be briefly described and the relationship among all will be illustrated.

The **Execution process** defines the complete development life cycle of the projects. This process is based on the waterfall model, as it is a sequential process where each phase should be completed before the next phase begins. A high level view of these execution phases is represented in Figure 4.3. Each phase has its own defined process that won't be described in detail, as it is out of the scope of this thesis.

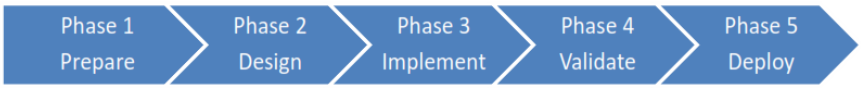


Figure 4.3 - Execution Process Phases, taken from [48]

The first phase concerns the preparation of the project and its team, by gathering the high level requirements that will enter in the PMP to be approved by the customer. Once the approval from the PM and the customer is obtained, the project enters in the design phase, which is the most demanding in terms of cost and schedule. It is in this phase that the requirements will be specified in detail, along with the test plans. After a peer review and the acceptance of the requirements specification by the customer, the implementation phase begins. In this phase, the developers will write the code of the solution and the unitary and integration tests will be performed. On the validation phase, system and acceptance tests will be executed and the project will be ready for acceptance. Finally, in the deploy phase, the delivery packages can be build and handover to the client or to maintenance.

The **Project Management process** takes the project through several stages, providing an understanding of the project's plans and making the progress visible to all stakeholders. This process complies with the rules of the project planning process area required at CMMI Level 2. The decision gates representing the major milestones of a project are illustrated in Figure 4.4.

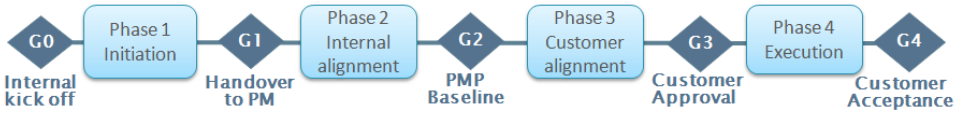


Figure 4.4 - Project Management Decision Gates, adapted from [49]

At gate 0 the project is handed over from sales to the Solution Center and the first phase begins. The objectives of the project will be set, so that on gate 1 the Delivery Manager can appoint a Project Manager based on his skills and experience. Then, the internal alignment phase begins to assure the commitment of the team to the project plans. After gate 2, the PM is responsible to assure that also the customer agrees to the plans, so that the PMP can be approved at the kick-off meeting at gate 3. All the execution of the project is done after gate 3, until the working products are accepted by the customer at gate 4.

The Requirements Management process can be framed in those two processes, as illustrated in Figure 4.5. The requirements are gathered by the Business Manager before they are handed over to the PM at gate 1, where the preparation phase starts. After the analysis of the requirements list, commitment to the plans must be obtained from the team (gate 2) and from the customer (gate 3), so that the design phase can begin. In this phase, the requirements and test cases are specified. If the customer approves the requirements specification documents, the

implementation phase begins, followed by validation and deployment phases. Finally, the work products are accepted by the customer at gate 4 and the project is handed over to maintenance.

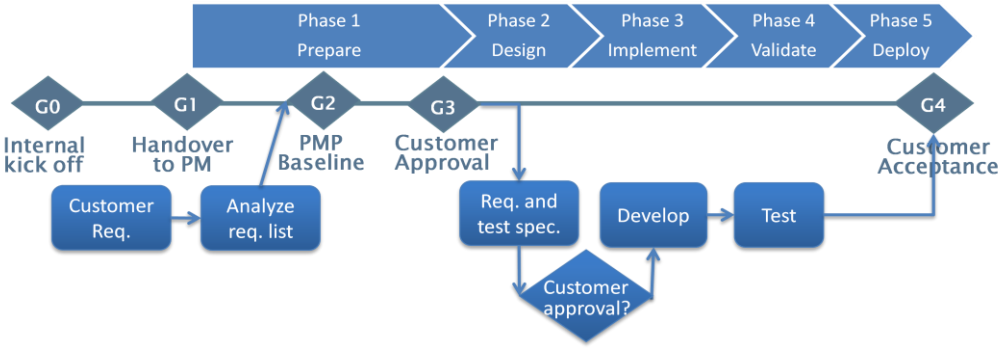


Figure 4.5 - Relationship between Requirements Management, Project Management and Execution processes

The set of processes previously presented respect the CMMI Requirements Management process area best practices. This compliance will be demonstrated in the next Section.

4.4. Gap Analysis

The compliance assessment presented in this section will be applied to two CMMI process areas. The REQM process area will be evaluated to demonstrate how the processes explained in the previous section meet the terms of CMMI. Also, the RD process area will be evaluated to determine the gaps between the current methodologies and the best practices required to achieve the desired capability level. Therefore, the SGD processes need to be evaluated regarding the goals and practices of both process areas. This analysis can be done through the identification of the processes' weaknesses and strengths and will determine where improvements can be made.

Such assessment of an organization is commonly referred to as **Gap Analysis**. According to the Business Dictionary [50], Gap Analysis is a technique used to determine what steps need to be taken to move a business from its current state ("what is") to its desired, future state ("what should be"). It consists of gathering the characteristics of the current situation, the factors needed to achieve future goals, and then highlight the gaps between both.

Gap Analysis is typically applied as a preparation for SCAMPI appraisals. Since the purpose of this assessment is not to generate a maturity rating, the few resources used and due to constraints of scope, time, training and experience this will be an informal assessment based on SCAMPI C rules.

4.4.1. Planning the Assessment

Determining ratings of maturity through goal satisfaction is permissible only in a SCAMPI A appraisal [37]. Therefore, similarly to the work of Espinheira [51], this assessment will be performed using the low-level CMMI elements. The goals will be evaluated using its associated practices, which will in turn be measured by its subpractices, since they provide guidance for interpreting and implementing the expected practices.

As a data structure to log the information collected during the assessment, the use of Practice Implementation Indicators (PIIs) is required [37]. PIIs provide a structure for relating each element in the model to individual pieces of objective evidence. The PII template used for this assessment is represented in Table 4.9.

Table 4.9 - Template of PII used for the Assessment

Goal	Practice	Sub-practice	Rating	Objective Evidence
SG 1	*SP 1.1*	...	✓	...
	SP 1.2	...	±	...
		...	×	...

The objective evidence collected is information that supports judgments made during the appraisal and can be classified in three types: direct artifacts, indirect artifacts and affirmations. The collection of this data was accomplished through interviews and review of documentation, such as the processes and templates used at Altran. The requirements documents of five projects were also analyzed. The projects *LESS*, *Anacom*, *AltranREQ*, *WSpace* and *ISINOV* were provided by Altran and were developed by three different Project Managers.

According to the SCAMPI B & C documentation [37], the characterization scale used to rate the practices must be a three-point scale. In this assessment, the scale generally used in SCAMPI B appraisals was applied to rate the subpractices, as defined in Table 4.10.

Table 4.10 - Required Scale for SCAMPI B, based on [37]

RED	The intent of the model practice is judged to be absent or poorly addressed in the set of implemented practices; gaps or issues that will prevent goal achievement, if the deployment occurred in this way across the organizational unit, were identified.
YELLOW	The intent of the model practice is judged to be partially addressed in the set of implemented practices; some gaps or issues were identified, which might threaten goal achievement if the deployment occurred in this way across the organizational unit.
GREEN	The intent of the model practice is judged to be adequately addressed in the implemented set of practices examined, in a manner that would support goal achievement, if the practice were deployed across the organizational unit.

In addition to the above, a designation of "Not enough evidences" can be used when no rate can be assigned because not enough data was gathered to rate the subpractice. If a subpractice didn't seem to be applied by any process, template or in any of the 5 analyzed projects, then the red label was assigned. The yellow label was given when at least one project provided evidenced of the subpractice, indicating that is not performed in a systematic way. If the subpractice is explicit in the template, in the process or in all analyzed projects, then the green label was assigned.

The rating of the specific practices is then derived according to the percentage of subpractices applied. Computing the percentage rate of each practice would be done as follows:

$$\text{Practice Rate \%} = \frac{1 * \text{number of green subpractices} + 0{,}5 * \text{number of yellow subpractices}}{\text{Total subpractices}} * 100$$

4.4.2. Requirements Management Assessment Results

In his section the Requirements Management process area will be evaluated to demonstrate that Altran's processes meet the terms of CMMI. Each of its specific practices (see Table 2.3) will have their subpractices rated and justified accordingly.

Table 4.11 - Rating of the subpractices of Requirements Management SP 1.1

Practices	*Sub-Practices*	*Rating*
1.1 Understand Requirements	1. Establish criteria for distinguishing appropriate requirements providers.	±
	2. Establish objective criteria for the evaluation and acceptance of requirements.	✓
	3. Analyze requirements to ensure that the established criteria are met.	✓
	4. Reach an understanding of the requirements with the requirements provider so that the project participants can commit to them.	✓

The full PII table for Requirements Management process area, providing all the subpractices' ratings and objective evidence supporting it, is available in Appendix C.

Practice SP 1.1 is accomplished by Altran through several activities. The requirements are registered on TestLink tool, along with their test cases, to ensure subpractice 2. The subpractice 4 is employed by Altran at gate 2 and 3 of the project planning process, as meetings with the project team and customer are conducted to obtain commitment to the project plans from both. The analysis of requirements (subpractice 3) is done at a high level in activity 2 of requirements

management and again after the refinement of the requirements during a review meeting in the design phase. Subpractice 1 was rated with the yellow label because it was considered there is room for improvement. Although the stakeholders are identified in the project plans, they are no clearly stated in the requirements documents generated.

Practice SP 1.2 aims at obtaining commitment to requirements. Altran fulfills this practice at gates 2 and 3 of the Project Planning process, as meetings with the team and customer are conducted to obtain commitment to the project plans. Additionally, subpractice 2 is also attained with the two gates of formal customer approval of requirements in the REQM process.

Practice SP 1.3 aims at managing changes to the requirements. The four subpractices are applied successfully by Atran's processes. Subpractice 1 is met as the requirements are documented in TestLink and in requirements specification documents. When requirements change, these changes are formally proposed in change requests, so that their impact on the project can be evaluated (following a risk Management process) and later formally approved by the project manager. All this activities are recorded to fulfill subpractices 1, 2, 3 and 4.

For **practice SP 1.4**, the TestLink tool is used to record the requirements and its dependencies, generating a traceability matrix of dependencies among requirements and between requirements and test cases. This fulfills all its subpractices.

Practice SP 1.5 ensures that project plans and work products remain aligned with requirements. In the Execution process, there are peer reviews and resolving issues activities at the end of each phase, which would fulfill subpractices 1, 2 and 3. However, these peer reviews are being neglected due to time constraints and the size of the teams. The milestone gates of the project management process help to track the progress of the project and ensure this alignment required.

The measurement of the subpractices allowed estimating the rate of the practices by the percentage of subpractices that were met by Altran's processes. The result is shown is Table 4.12.

Table 4.12 - Percentage rate of the Requirements Management specific practices

GOALS	PRACTICES	Rating
SG 1 Manage Requirements	SP 1.1 Understand Requirements	87,5%
	SP 1.2 Obtain Commitment to Requirements	100%
	SP 1.3 Manage Requirements Changes	100%
	SP 1.4 Maintain Bidirectional Traceability to Requirements	100%
	SP 1.5 Ensure Alignment between Project Work and Requirements	50%

The results of this gap analysis demonstrate (as expected for a company rated with Maturity level 2) that the current processes do comply with the best practices of REQM process area. Nevertheless, there is room for minor improvement, specifically in practice SP 1.1 by stating stakeholders more clearly in the Requirements documentation and practice SP 1.5 since the peer reviews are not being carried out like stated in the execution process. Figure 4.6 summarizes the strengths and weaknesses found.

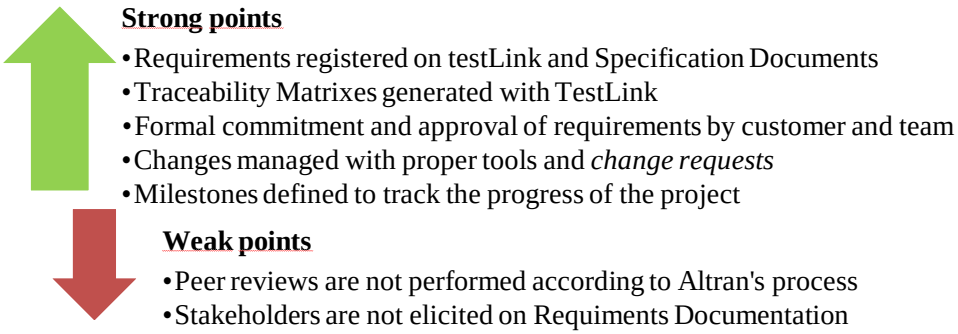


Figure 4.6 - Diagram of the main strengths and weaknesses of the current processes

4.4.3. Requirements Development Assessment Results

The Requirements Development process area requires the satisfaction of 3 specific goals (see Table 2.4). In this section, this process area will be evaluated in order to identify the areas that need improvement to reach the desired capability level. Each of its goals will have their practices evaluated through the rating of their subpractices.

The full PII table for Requirements Development process area, providing all the subpractices' ratings and objective evidence supporting it, is available in Appendix C.

Table 4.13 - Rating of the subpractices of Requirements Development SP 1.1 and SP 1.2

Practices	*Sub-Practices*	*Rating*
1.1 Elicitation of Needs	1. Engage relevant stakeholders using methods for eliciting needs, expectations, constraints, and external interfaces.	✓
1.2 Transformation of needs into Customer Requirements	1. Translate stakeholder needs, expectations, constraints, and interfaces into documented customer requirements.	✓
	2. Establish and maintain a prioritization of customer functional and quality attribute requirements.	×
	3. Define constraints for verification and validation.	✓

The **first goal** contains two practices. At Altran, the elicitation of needs, expectations and constrains is primarily done by the BM in the first meeting with the customer through interviews and document analysis. If needed, the Project Manager can also gather more requirements with the customer, so they can be analyzed and translated into the documented Customer Requirements List included in the PMP. These activities fulfill the first subpractices of each practice. The constraints of subpractice 3 are realized through the test cases associated to each requirement. Subpractice 2 of SP 1.2 is not currently performed at Altran.

Practice SP 2.1 introduces the notion of product component. The term "Module" is the common terminology used at Altran to denote a product component. The first subpractice is performed in the Requirements Specification activity of REQM process, specifically using a technical design specification template. Subpractice 4 is also accomplished as requirements are registered in TestLink, along with their dependencies. The derivation of news requirements of subpractice 2 is performed, however the rationale behind those decisions is not registered and the review of the requirements after the technical design specification is unstated. Subpractice 3 is not performed at Altran as the quality attributes are not captured systematically and they don't directly drive the design decisions of the project.

In the requirements specification document and in TestLink, the requirements are grouped according to modules, so subpractice 1 of **practice SP 2.2** is met. Subpractice 2 is also met because the requirements are allocated to the architecture in the technical design specification. The design constrains are currently blended with requirements, so subpractice 3 is not met. According to Atran's Requirements Management process, subpractice 4 is done during the analysis of the requirements list for the PMP. Subpractice 5 is aligned with subpractice 4 of SP 2.1 and therefore is also addressed.

The identification of interface requirements (**practice 2.3**) is currently performed. However, subpractice 1 is not fully met as only the external interfaces are considered. The interface requirements are developed as they are registered in TestLink along with their dependencies.

Practice 3.1 is not applied at all at Altran as there are no scenarios or operational concepts defined for the project. The definition of the environment is suggested in the technical design specification document, but is not systematically done in all projects.

Practice 3.2 considers the definition of functionality through Functional Analysis and the impact of quality attributes on the architecture. Altran is currently documenting some quality attributes. However, their impact on the architecture is not assessed, nor derived from the key business drivers. Therefore, subpractice 2 is partially met and subpractice 3 is not performed at all. As for the functionality, Altran defined a functional design template where the process flows, inputs, outputs and their responsibilities are described. This meets subpractice 5 and

partially meets subpractices 2 and 4, although the functionality is not measured. As mentioned before, subpractices 6, 7 and 8 are accomplished since the requirements are grouped in TestLink according to their modules (similar functionality). The key business drivers of subpractice 1 are identified by the Business Manager during the proposal.

The analysis of requirements (**practice 3.3**) is done for the first time after the Project Manager receives the requirements from the BM. After the requirements specification activities, there are review meetings to analyze the completeness and consistency of requirements. These activities fulfill subpractices 1, 2 and 3. Subpractices 4, 5 and 6 are not currently performed.

Practice SP 3.4 suggests 4 practices that are related to requirements risk analysis. The Risk Analysis is not yet performed specifically for the requirement and therefore subpractice 2 and 3 are not met. Subpractice 4 is done at a very high level by the Practice Manager when he proposes the solution. The balance of stakeholder needs and constraints in subpractice 1 is partially accomplished as the prototypes are presented in the documentation, but this subpractice requires a negotiation with the Customer that is not performed.

The validation of requirements (**practice 3.5**) is somehow performed as the customer needs to approve the requirements specification document, an activity established by the requirements management process. In this document, the prototypes of the interfaces are presented. This approval can be done with the customer via email, and therefore the feedback received by all the relevant stakeholders is constrained (subpractice 2). For the same reason, subpractice 3 is partially performed. The assessment of design as it matures is not done systematically after the meeting with the customer. Subpractice 1 is not performed by Altran.

The measurement of the subpractices allows estimating the score of the practices by the percentage of subpractices that were met by Altran's processes. The result is shown in Table 4.14.

Table 4.14 - Percentage rate of the Requirements Development specific practices

GOALS	PRACTICES	Rating
SG1 Develop Customer Requirements	SP 1.1 Elicit Needs	100%
	SP 1.2 Transform Stakeholder Needs into Customer Requirements	67%
SG2 Develop Product Requirements	SP 2.1 Establish Product and Product Component Requirements	63%
	SP 2.2 Allocate Product Component Requirements	80%
	SP 2.3 Identify Interface Requirements	75%
SG3 Analyze and Validate Requirements	SP 3.1 Establish Operational Concepts and Scenarios	12%
	SP 3.2 Establish a Definition of Required Functionality and QA	75%
	SP 3.3 Analyze Requirements	50%
	SP 3.4 Analyze Requirements to Achieve Balance	25%
	SP 3.5 Validate Requirements	33%

This diagnose showed that there are significant gaps between the current processes used by Altran and the CMMI Requirements Development expected practices. Figure 4.7 depicts the major strengths and weaknesses of the current methodology.

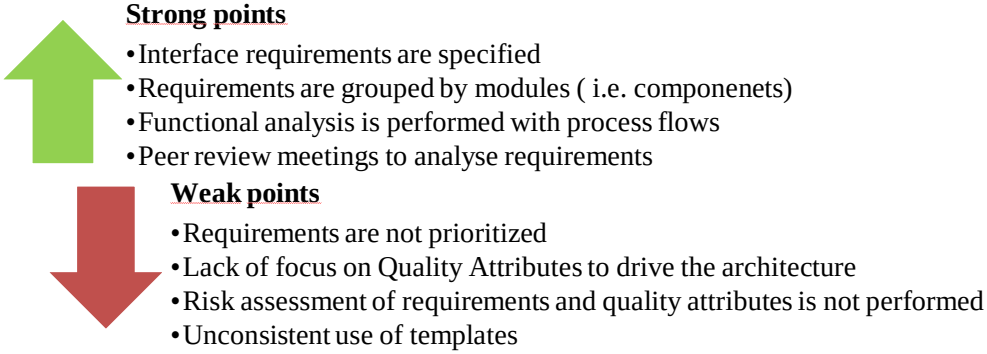


Figure 4.7 - Diagram of the main strengths and weaknesses of the current processes

Although not mandatory to satisfy the goals, the transformation of stakeholders needs into customer requirements could be enhanced with the prioritization of requirements. Also, more emphasis should be given to the quality attributes that drive architectural design decisions. The risk assessment of the requirements and quality attributes on the project is not yet performed at Altran. Thus, this is another issue that could be improved to meet the goals of this process area. Finally, the analysis of the documentation generated by some projects allowed detecting that the templates currently available help meet some subpractices, but are not used systematically. This happens because in a consulting company there is a great need for flexibility due to the variability of domains, technologies and clients. So, the templates are adapted to suit the needs of the Project Manager and team. However, to assure the practices of this process area, the variability of the template will have to restrict with mandatory sections. The aim will be on maintaining the flexibility but with defined rules to do it.

4.5. Improvement Plans

The improvement opportunities detected in the previous section helped to derive four research questions that will guide the investigation of alternative solutions for each gap found. The questions are presented in Table 4.15, however such a study will only be performed during the next phase of this dissertation.

Table 4.15 - Research Questions for the study of alternative solutions

Question	*Rationale*
Q1 *What are the main techniques used to prioritize requirements?*	The answer delivers an overview of what are the main techniques used to prioritize requirements. This will allow the company to accomplish practice 1.2 of RD process area.
Q2 *How to derive the architecture using Quality Attributes?*	This question will help to fulfill practice 2.1 and 3.2 by revealing how the architectural decisions are related to quality attributes.
Q3 *What are the main techniques to analyze the risk of requirements?*	The answer will reveal the main techniques to reasoning about risks during the requirements analysis process. It will help to meet practices 3.4 and 3.5.
Q4 *How to define scenarios and operational concepts?*	This question discovers how to define scenarios and operational concepts in order to improve practice 3.1.

4.6. Summary

This chapter offers an overview of Altran's processes and their compliance with CMMI.

Their Delivery Management System (SGD) consists of a collection of CMMI-compliant processes with activities, inputs, outputs, responsibilities, rules and other process elements that guide the development and maintenance projects of the Solution Center. In particular, the Requirements Management process was detailed, regarding its main roles and responsibilities, the activity workflow and tools used to assist this process. Since Requirements Management process can be contextualized in the other processes of the SGD, the relationship between this and other relevant processes is also briefly described and illustrated.

To conclude, an evaluation method for a quick gap analysis was defined. This assessment, based on an informal SCAMPI C, was conducted to demonstrate the compliance of the current methodology with the RM process area and to highlight the improvements needed to conform with the RD process area. Using the outcomes, 4 research questions were derived to guide the future investigation of solutions. The results of this assessment were then validated through interviews with Project Managers. Their opinions on the main problems of the current processes were gathered, indicating that the main problem is related to the overhead in bureaucracy, which has already been described has an intrinsic problem of CMMI (refer to Section 2.2.4).

5. Investigation of Alternative Solutions

This chapter answers the four research questions defined in Section 4.5 to guide the investigation of alternative solutions for the main gaps found – Section 5.1 uncovers the main techniques used to prioritize requirements; Section 5.2 studies the influence of quality attributes in the architecture; Section 5.3 explores the main techniques to analyze the risk of requirements and Section 5.4 investigates how define scenarios and operational concepts. Each question will be examined independently, based on academic research and empirical surveys found in the literature. To conclude this Chapter, the solutions chosen for Altran are summarized in Section 5.5. Such solutions will be the basis for the new Requirements Management and Development Methodology for Altran.

5.1. RQ1: What are the main techniques used to prioritize requirements?

Requirements Prioritization is an important activity of Requirements Engineering. Sommerville defines this term as '*the activity during which the most important requirements can be discovered*'[52]. This activity can be seen as a complex multi-criteria decision making process performed during the requirements negotiation and release planning [53].

Given that most projects are challenged with cost, time and resource constraints, the prioritization of requirements is used to maximize business value by deciding the order of implementation of requirements and group them in several releases [54], [55]. Therefore, Requirements Prioritization is a critical success factor for perceiving the stakeholders' genuine needs and ensuring customers' satisfaction [53]. It helps software engineers to plan staged deliveries, but also to resolve conflicts and negotiate trade-offs, achieving an agreed and consistent view of the system.

Several criteria for obtaining this prioritization exist. The stakeholders' preference, the time, cost, penalty, and risk of requirements are examples of such criteria that can be considered when deciding the requirements implementation order [56]. According to [55], the *customer input* is the most common criteria used in industry, but absence of criteria is also frequent.

5.1.1. Requirements Prioritization techniques

Regarding the available prioritization techniques, Achimugu *et al.* conducted a Systematic Literature Review where 49 prioritization techniques were identified [53]. Such techniques can be classified in three main categories:

- **Nominal Scale techniques** focus on prioritizing requirements by dividing them into defined groups of priorities. Requirements in one group are of equal priority.
- **Ordinal Scale techniques** are based on an ordered list of requirements.
- **Ratio Scale techniques** present the relative difference between requirements' importance.

For the purposes of this dissertation, we selected five prioritization techniques for analysis, based on their popularity in academia and acceptance in industry. While the Analytic Hierarchy Process **(**AHP) was by far the most cited in research papers [53], Numerical Assignment and ad-hoc techniques were pointed out as the most used in industry [55]. This contrast relies on the fact that complex and time consuming techniques are usually avoided in industrial practice.

5.1.1.1. Nominal Scale techniques

Numerical Assignment

Description: *Numerical Assignment* is a simple prioritization technique based on a nominal scale, which aims at grouping requirements into different priority groups. The number of groups can vary, but typically three priority groups are used (e.g.: "Critical", "Standard" and

"Optional"). The requirements contained in the same group have equal priority, and for that reason, it is not possible to measure the relative differences between requirements within a group.

Strengths: The simplicity of the technique. It is very easy to use and fast to perform.

Weaknesses: Different stakeholders might have different interpretations of the groups and might place the majority of requirements in the "critical" group, threatening the overall goal of prioritizing the most important requirements. In addition, since all requirements within a group are at the same priority level, it is not possible to measure their relative differences. However, this problem can be avoided by applying other techniques (e.g., *Ranking*, *Cumulative Voting*) to the requirements in each single group.

Variations: *MoSCoW* is a kind of *Numerical Assignment*, which groups requirements in four priority groups ("Must", "Should", "Could", "Wont").

5.1.1.2. Ordinal Scale techniques

Ranking

Description: *Ranking* is a simple and intuitive prioritization technique based on an ordinal scale, which numerically ranks all requirements without any ties. Considering we have *n* requirements to prioritize, number 1 is assigned to the most important requirement and *n* to the least important one. Unlike *Numerical Assignment*, each requirement has a unique rank. However, it is not possible to measure the relative difference between the ranked items as in *AHP* or *Cumulative Voting*.

Strengths: The simplicity of the technique. It is easy to use and rather fast to perform.

Weaknesses: It is hard to apply when there are multiple stakeholders, as there might be conflicting viewpoints and ties in priorities. Also, it is not possible to measure the relative difference between the ranked requirements. Moreover, as the number of requirements increases, it gets harder to remember all the requirements.

Variations: The ranking can be performed using several sorting algorithms, like *Bubble Sort*, *Quick Sort* or *Binary Search Tree*. However, this increases the complexity and time to perform the process, making the technique harder to use.

5.1.1.3. Ratio Scale techniques

Cumulative Voting

Description: *Cumulative Voting* (also known as *100$ Test*) is a relative simple prioritization technique based on a ratio scale. In this technique, each stakeholder is given 100 imaginary units (e.g., money, importance or hours) that he can distribute across all requirements. Because requirements weights' are not assigned in a sequential order, the results provide information about how important a requirement is relative to the others.

Strengths: The technique is relatively easy and fast to use, and provides information about the relative differences among of requirements' importance.

Weaknesses: For large numbers of requirements, it gets harder to apply as stakeholders might miscalculate the sum. However, the use of automated tools may prevent this problem. Also, it is possible for stakeholders to influence the results if they put all their units in a favorite requirement, perceived as low priority by other stakeholders. This problem can be avoided by limiting the amount of units in a single requirement, but would also prevent stakeholders to prioritize according to their own will.

Variations: Using values other than 100 units.

Analytic Hierarchy Process (AHP)

Description: *Analytic Hierarchy Process* (*AHP*) is a structured decision-making technique to organize and analyze complex decisions, based on mathematics and psychology [57]. This technique was proposed by Saaty in the 70's and is used in a variety of domains. It can be used to prioritize requirements by reducing complex decisions to a series of one-on-one comparisons between all possible pairs of hierarchical requirements. A numerical priority is calculated for each element of the hierarchy and a consistency ratio calculated to ensure the validity of results.

Strengths: It has been a widely researched method and provides the most reliable results. Also, it provides information of the relative importance between requirements.

Weaknesses: It is the most hard to use and time consuming technique, with a complexity of $O(n^2)$. Therefore, it also suffers from scalability problems as the number of requirements increases.

Variations: *Pair-wise Comparison*, *Hierarchy AHP* and *Cost-Value approach* are prioritization techniques based on AHP that aim at reducing the number of comparisons needed. However, by reducing the comparisons, there is a risk of decreasing the technique reliability.

5.1.1.4. Mixed techniques

Planning Game

Description: *Planning Game* is a technique used in eXtreme Programming that combines *Numerical Assignment* and *Ranking*. In this technique, requirements (written as story cards) are first prioritized into three groups and then are ranked within each group. It requires the participation of both customers and programmers, which should cooperate to create a release plan that will maximize business value while minimizing the costs. Since customers have the most information about value they are responsible for grouping the requirements in three groups of importance: "those without which the system will not function", "those that are less essential but provide significant business value" and "those that would be nice to have". At the same time, programmers should estimate the costs of each requirement and group them according to these three groups of risk. Based on their estimates, the customers are then responsible for sorting the grouped requirements and select which should be planned for the next release.

Strengths: This technique is rather flexible and can scale up to large number of requirements (O(n) complexity) with reasonable effort and time.

Weaknesses: Since the result is an ordinal scale, there is no information about how important one requirement is relative to others.

Variations: *PGcAHP* combines *Planning Game* with *AHP*.

5.1.2. Comparative Analysis

To understand which prioritization techniques best suit Altran's needs, these five selected techniques were analyzed using the findings of several existing comparison studies ([53], [56], [58]–[61]). The results of such studies were then integrated, using a comparison criteria based on Kahn's research framework [58]:

- *Ease of use:* Measures how easy a particular prioritization method is to perform.
- *Time:* Measures the time spent to complete the prioritization process.
- *Reliability*: Measures how reliable the result of a particular prioritization technique is, i.e. how sensitive it is to the insertion of judgment errors.
- *Scalability*: Measures how many requirements could be prioritized with a reasonable effort, or how difficult it becomes to use with increasing the number of requirements.
- *Changeability*: Measures the ability of a technique to automatically update ranks as requirements evolve, with inserting new requirements and deleting existing ones.

Each technique was evaluated against the aforementioned criteria. The outcomes were assembled in Table 5.1 and classified according to the following scale:

- ☺ – The technique highly satisfies the evaluated criteria.
- 😐 – The technique moderately satisfies the evaluated criteria.
- ☹ – The technique does not satisfy the evaluated criteria.

Table 5.1 - Summary of the analysis of the selected techniques

Prioritization Technique	Category	Ease of use	Time	Reliability	Scalability	Changeability
Numerical Assignment	Nominal	☺	☺	☹	☺	☹
Ranking	Ordinal	☺	☺	☹	☹	☹
Cumulative voting	Ratio	😐	😐	😐	☹	☹
AHP	Ratio	☹	☹	☺	☹	☹
Planning Game	Mixed	😐	😐	😐	☺	☹

The results show that although *AHP* is the most investigated technique and has the most reliable results, it takes the longest time consumption and effort due to the large amount of decisions it requires. Therefore, like most methods, *AHP* suffers with scalability issues because it becomes impracticable to compare all requirements pairs. As the number of requirements increases, the number of relative priorities comparisons increases with a magnitude of $O(n^2)$, requiring four times the effort or time consumption [53], [58]. Moreover, the studies showed that *AHP* is very complex in comparison to the other techniques and therefore is the hardest to use [59]–[61].

Numerical Assignment and *Ranking* were indicated as the easiest techniques to use [56], [60]. They take the less time to perform and provide high degrees of user confidence. Nevertheless, they provide the less reliable results. It is also interesting to note that *Numerical Assignment* seems to be the one of the few methods that suits large numbers of requirements [60]. However, since requirements that are in the same priority group represent equal priority, this technique cannot provide relative differences between these requirements priorities and is not as meaningful as ordinal or ratio scale methods.

The *Cumulative Voting* technique takes longer to perform than *Numerical Assignment* and Ranking, but it is still relatively easy to use [60]. Similarly to *AHP*, the *Cumulative Voting* and *Ranking* techniques also contain scale up problems due to the difficulty that people may feel when remembering a large number of requirements [62]. In particular, in *Cumulative Voting*, the stakeholders may assign the same priority to many requirements, especially the requirements with low priorities.

The *Planning Game* was indicated in [61] as a rather fast and easy to use prioritization technique, which presented better empirical results than *Cumulative Voting*. Given that its computational complexity is O(n) comparisons, it is also less problematic and flexible enough to scale up to higher numbers of requirements.

Regarding the changeability, none of the selected techniques seems to easily update the priority rankings as requirements evolve. According to [53], this is a limitation inherent to most existing prioritization techniques.

5.1.3. Discussion

Requirements Prioritization has been a considerably discussed subject by the research community. Although many prioritization techniques exist, improvements are still required. The lack of scalability, the difficulty to accommodate requirements changes in rank updates, the coordination among stakeholders and requirements dependency issues are the main limitations of the current prioritization techniques [53].

Different situations require different prioritization mechanisms and none can be considered the best one [63]. While complex techniques like AHP are very reliable and useful when sensitive

analysis is needed for a small number of requirements, simpler techniques might ensure cost effective decisions and are generally preferred by the industry [55], [56], [58]. In fact, a combination of techniques is also possible, for instance, ordinal scale methods can be used to complement the performance of the nominal scale ones [53].

It is also interesting to note that most of the studies found comparing prioritization techniques generally used less than 20 requirements [63]. In her systematic literature review [54], Ma concluded that, the strength of evidence for the effectiveness of prioritization techniques for medium and large number of requirements is weak and more studies are needed. This might also indicate that it is less valuable and more time consuming to prioritize on lower levels as compared to higher levels [58].

The previously presented techniques can be used with both functional and non-functional requirements. However, Berander *et al.* stated that it is not always possible or advisable to prioritize both types of requirements together due to its intrinsic differences in nature [56]. Still, this is not what happens in practice. In fact, during an empirical study on how Quality Requirements are prioritized in industry [55], Svensson *et al.* concluded that simple techniques like *ad-hoc* and *Numerical Assignment* are the dominant for prioritizing both Functional and Quality requirements. In addition, they also discovered that Quality Requirements are by default seen as having a lower priority than Functional Requirements, and only prioritized if time and resources are available. One explanation for the lack of focus on prioritizing this type of requirements might not be related to the prioritization process itself, but rather on the lack of practitioners' knowledge on how to manage non-functional requirements in the overall RE process.

5.2. RQ2: How to derive the architecture using quality attributes?

The second research question aims at satisfying a subpractice of SP 2.1 that aims to *"Develop architectural requirements capturing critical quality attributes and quality attribute measures necessary for establishing the product architecture and design"*.

Quality attributes are a subset of Non-Functional Requirements (NFRs). The IEEE standard Glossary for Software Engineering defined the term as 'a *feature or characteristic that affects an item's quality'* [3]. They affect different activities and roles throughout the software development life cycle. In particular, quality attributes should be addressed as early as possible and properly reflected in the architecture before committing to a design [64], [65]. In fact, it has been acknowledged that quality attributes and constraints have a strong influence in Software Architecture, being the main drivers of architectural decisions [65]. Software architects should use them as a selection criterion among alternative designs and implementations.

Clements *et al.* [65] also recognized this tight relation between quality attributes and the architecture and classified them in the three following classes:

- **System Qualities**, such as availability, modifiability, performance, security, usability.
- **Business Qualities**, such as the time to market, cost and benefit.
- **Architectural Qualities**, such as conceptual integrity, correctness, completeness.

Empirical studies on the subject of quality attributes [66], [67], reported that Performance and Usability were considered the two most important System Qualities. Maintainability and Security were also regarded as essential, however most respondents considered them as common sense characteristics of any system whose satisfaction might be delegated to the technologies used [66]. On the other hand, it was also discovered that Business Qualities such as licensing issues, technological policies or the cost are also considered as relevant as system qualities. However, this perceived importance given to quality attributes is not always the same. It depends on several factors such as the application domain, the type and size of the project, or the stakeholders' specific needs [67].

These quality attributes often relate or may conflict with each other due to their nature or different stakeholders' priorities and expectations. Therefore, trade-offs need to be made through informed decisions to find the combination that better satisfies the stakeholders goals. The basic question that architects should ask themselves when prioritizing these quality attributes is "*How much must I give up to get a little more of what I want the most?"* [68].

In industrial practice, quality attributes are also considered important and, recently, further resources are being invested to achieve them [66]. Still, there seems to be a gap between theory and practice as the functionality is always prioritized and documented over them [4] [6]. Besides, most companies don't have a differentiated role for Software Architects. Usually, they accumulate other roles in the project (such as project manager or developer) and make architectural decisions based on their own experience and knowledge, rather than their skills as architects [66].

According to a survey conducted by Ameller *et al.* [69], these quality attributes are the drivers of architectural decisions, which can be categorized in four main types:

- **Architectural patterns:** Quality attributes can influence the preference for certain architectural patterns. For instances, a layered architecture is a common choice to support latter changes.
- **Implementation strategies:** General or detailed design decisions can help to satisfy certain quality attributes, like the use of duplicated tables of a database to decrease access time.
- **Transversal decisions:** Decisions that affect the whole architecture. For example, the use of third- party components or open source software.
- **Technological platforms:** Decisions related to technological choices in the database, middleware, etc. For example, the use of Oracle databases is a common choice when aiming for high availability.

From the Requirements Engineering perspective, quality attributes need to be elicited, documented and validated throughout the development process, so they can be used to drive such architectural decisions.

Quality Attributes are typically difficult to elicit. Studies show that, it is often the architect himself that elicits them based on his experience [66], [69]. Although they are derived from the customer's business goals, they only mention them in a broad way or don't bring them up at all. Moreover, there are often terminology confusions and communication problems, especially with Quality Attributes definitions, as architects don't share a common vocabulary [66]. The elicitation process is considered to be iterative and expanding along the system lifecycle.

The documentation of quality attributes represents a major gap between academics and practitioners [66]. Although several methods and frameworks have been proposed to represent and model them (like the NFR Framework [19] or Volere Templates [70]), their documentation is often neglected or imprecise. Therefore, the documentation of quality measures is also scarce.

Since the documentation of quality measures is poorly maintained, their validation also becomes difficult to perform [66][69]. Besides, the validation technique used is highly dependent on the quality attribute being evaluated. For instances, to validate performance, stress tests can be employed, while for usability simple prototypes are feasible.

To automate these activities, some Non-functional Requirements Management tools have been proposed by the research community to model and analyze NFRs trade-offs. However, in practice, software architects don't seem to use any of them [66].

5.2.1. Methods, Techniques and Tools

Even if quality attributes seem to be addressed in an *ad-hoc* fashion by the companies, some methods, techniques and tools exist to represent, analyze trade-offs and derive architectural decisions from them.

5.2.1.1. The NFR Framework

The NFR Framework [19] is the most recognized process-oriented approach proposed in academia. It is a Goal-Modeling framework used to represent non-functional requirements, analyze trade-offs and the possible operationalizations that can satisfy them at the functional level. Quality attributes are represented as *softgoals* that can positively or negatively contribute towards each other and that are recorded in a *softgoal* interdependency graph. These *softgoals* can be hierarchically decomposed and refined in a tree structure, in which the leaf goals are at a granularity level that can be operationalized. The possible operationalization alternatives are selected and their impact is propagated up the NFR refinement tree to analyze the satisfaction of the parent softgoals. An example of such a *softgoal* interdependency graph is represented in Figure 5.1.

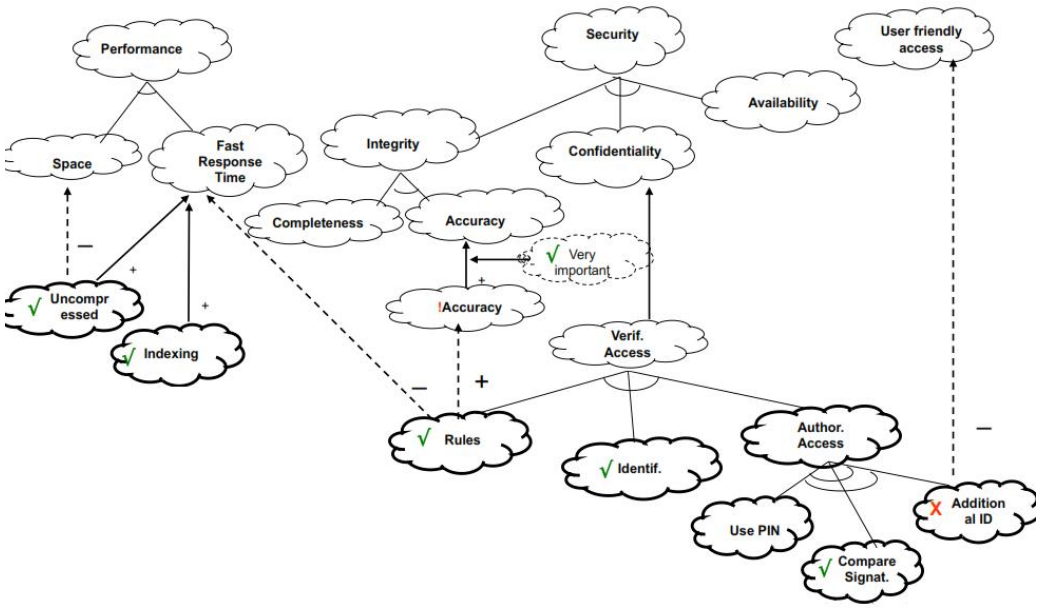


Figure 5.1 - Softgoal Interdependency Graph for security, taken from [17]

It is interesting to note that an operationalization that satisfies one softgoal can also harm another one. For example, the choice of rules to verify access, aids in accuracy but hinders the response time. To understand how the set of operationalizations chosen will impact on the quality attributes, a bottom-up evaluation procedure can be propagated from leafs to top goals, considering positive/negative contributions and AND/OR decompositions.

Besides recording and documenting quality attributes, this framework helps to justify design decisions during the software development process and allows reusing the quality attributes knowledge gathered through NFR catalogues. Several catalogues of non-functional requirements types have already been proposed, however a central repository where then can be consulted doesn't currently exist.

5.2.1.2. Process to handle NFRs in the context of Software Architecture

To systematically select architectural styles and patterns from quality attributes, a generic process to handle NFRs in the context of Software Architecture was proposed by Moreira [71]. The main activities of this process are illustrated in Figure 5.2.

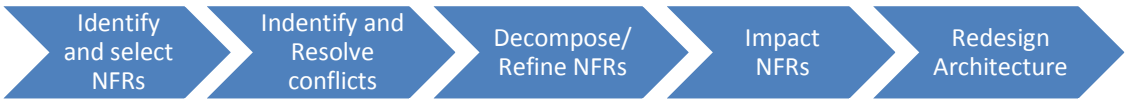


Figure 5.2 - Proposed process to handle NFRs, adapted from [71]

The Requirements Engineer should start by identifying candidate non-functional requirements, selecting the most import ones for the project. The identified NFRs can either positively or negatively impact on each other. To identify such conflicts, a contribution matrix (see Table 5.2) is build using existing body of knowledge or the architect expertise.

Table 5.2 - Quality Attributes Contribution matrix

QA \ QA	Response Time	Availability	Security	Multi-Access
Response Time		+	-	-
Availability				+
Security				
Multi-Access				

As these desired qualities don't all relate to the same functionalities, a NFRs/Use case relationships matrix, for example, is also built (see Table 5.3). Other requirements artifacts could be used to reflect the different system functionalities in this matrix. From these two matrixes, conflicting quality attributes can be identified and trade-offs analyzed. Negative contributions on the same functionality are then assigned a priority (for example on a qualitative scale as *Very important*, *Important*, *Average*, *Not so important* and *Do not care*) based on the combination that best satisfies stakeholders' need.

Table 5.3 - QA vs. Use Case Contribution matrix (example)

QA \ UC	Use Case 1	Use Case 2	Use Case 3	Use Case 4
Response Time		**Very important**	**Very important**	
Availability		X	X	
Security	X			X
Multi-Access		**Average**	**Important**	

To decompose/refine NFRs into operationalizations and design decisions, the Softgoal Interdependency Graph of the NFR Framework previously explained can be used. The architect identifies possible catalogues or proposes possible refinements, and the selected operationalizations will then be added to the functional models and an NFR/Architectural Style matrix (see an example in Table 5.4) can be used to select the style that satisfies the most NFRs. An architectural style is a "*named collection of architectural design decisions that (1) are applicable in a given development context, (2) constrain architectural design decisions that are specific to a particular system within that context, and (3) elicit beneficial qualities in each resulting system*" [72].

Table 5.4 - QA vs. Architectural Style Contribution matrix (example)

QA \ AS	Shared Data	Abstract Data Type	Implicit Invocation	Piper & Filter
Modifiability	--	+	-	--
Space	++		-	--
Response Time		-	--	
Reusability	-	+	+	+

Although this last analysis is no longer a part of the requirements phase, it will have a direct impact in the architectural design. Therefore, it was considered in this research question for the sake of completeness. However, will not be part of the new process for Altran as it is outside of our scope.

5.2.1.3. Quality Attribute Scenarios

Clements *et al.* [65] proposed another approach based on scenarios to represent quality attributes and analyze possible architectural design decisions. Quality Attribute Scenarios can be represented with the following six parts:

- *Source of stimulus*: Entity (a human, a computer system, or any other actuator) that generates the stimulus.
- *Stimulus*: Condition that needs to be considered when it arrives at a system.
- *Environment:* Set of circumstances or system's state in which the stimulus occurs.
- *Artifact*: The stimulated artifact may be a collection of systems, the whole system, or some pieces of it.
- *Response*: Activity undertaken as the result of the arrival of the stimulus.
- *Response measure*: Response should be measureable so that the requirements can be tested.

These scenarios can be either general or concrete. *General scenarios* are system independent and each of the six parts contains a range of possible values. They are particularly useful to support the elicitation of quality attributes and act as a checklist to ensure all possibilities have been considered. Concrete, *system-specific scenarios* are derived from general scenarios by choosing one or more values from each part of the scenario. They are meaningful, possible to test and, therefore, they can be used as the quality attribute requirements for a system. The authors argue that '*concrete scenarios play the same role in the specification of QA requirements that Use Cases play in the specification of functional requirements*'. A general scenario and a derived concrete scenario to achieve performance are given in Table 5.5 and Figure 5.3, respectively.

Table 5.5 - Performance Generic Scenario, taken from [65]

Portion of Scenario	Possible Values
Source	One of a number of independent sources, possibly from within the system
Stimulus	Periodic events arrive; Sporadic events arrive; Stochastic events arrive
Artifact	System
Environment	Normal mode; Overload Mode; Emergency Mode
Response	Processes stimuli; Changes level of service
Response Measure	Latency; Deadline; Throughput; Jitter; Miss rate; Data loss

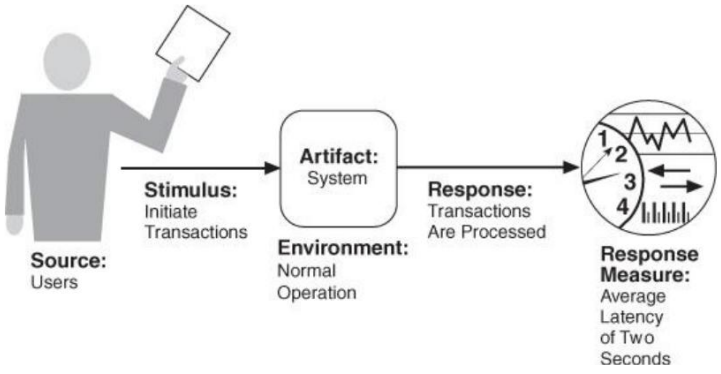


Figure 5.3 - Performance Sample Scenario, taken from [65]

The achievement of these quality attributes relies on the set of fundamental design options selected by the architect. Such design decisions, referred to as *tactics*, influence the control of quality attribute responses. Tactics that can be refined with other tactics are organized into a hierarchy, such as the example of Figure 5.4. A collection of tactics that supports certain quality attributes is an architectural pattern. They are valuable because they can be cataloged and therefore allow the architect to reuse knowledge.

This approach enables the communication between stakeholders by providing a common vocabulary and testable definitions of quality attributes. General Quality Attribute scenarios for *availability, modifiability, performance, security, testability, usability* and a list of tactics to achieve these quality attributes can be consulted in [65].These tactics are somehow equivalent to the operationalizations of the NFR Framework. For the characterization of other quality attributes, a number of taxonomies can be found in the literature. Nevertheless, organizations

are also encouraged to gather their own knowledge and create their own general scenarios and regular tactics to solve them.

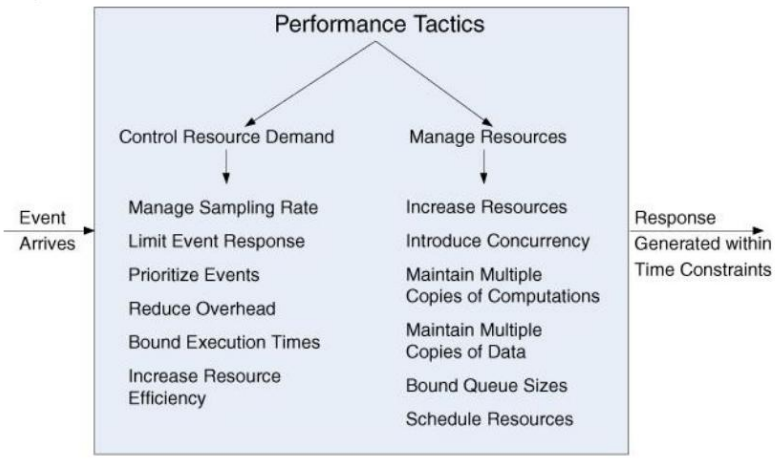


Figure 5.4 - Performance Tactics, taken from [65]

5.2.1.4. The Quark method

The Quark method is another approach, proposed by Ameller *et al.* [73], to facilitate the architect's decision making process with regard to quality attributes. It is based on an iterative process and relies on the Arteon Ontology [74] to represent architectural knowledge. The process starts with the quality attributes and constraints provided by the architect. Using the Arteon Ontology, it is possible to infer a list of architectural decisions, according to the available architectural knowledge and a prioritization criterion. After an ordered decision list is generated, the architect should select the most convenient decisions. Some incompatibilities between decisions can arise in this process. The Quark method is only responsible to detect and inform the architect about conflicts and dependencies among decisions or to give suggestions. It is up to the architect to either accept them or start a new iteration to solve the detected issues.

This method is a lightweight approach to support architects with architectural design decisions. Since it doesn't select or prohibit any decisions by itself, the method grants the architect with full control and a central role in the process. To automate this method, the authors developed an eclipse-based tool, ArchiTech [75], that used Quark and provides an overall evaluation of the given quality attributes. Nevertheless, the acquisition and management of the Architectural Knowledge still represents a problem as large networks of knowledge are required, but difficult to gather.

5.2.2. Discussion

The influence that QA have on architectural design decisions is a concern with growing interest. Some methods and techniques have been proposed in academia to deal with this issue. The four methods described in the scope of this research question have been chosen considering not only their popularity and acceptance in industry and research communities, but also for the distinct approaches they follow when attempting to solve the problem.

While the **NFR Framework** [19] adopts a goal-oriented approach to model quality attributes, their operationalizations, contributions and trade-offs, **Quality Attribute Scenarios** [65] are scenario-oriented and characterize quality attribute requirements based on the functionality and its measure. Both techniques are simple to use, however they do not determine how QA should drive the design decisions. Hereof, the **Quark method** [73] is more suitable since it uses a knowledge base to assist the architect in the decision making process based on previous experience. However, building such a knowledge base takes a lot of time, ontology awareness and input from several architects. Likewise, the **process to handle NFRs in the context of Software Architecture** proposed by [71] also supports the architects with design decisions as it

uses contribution tables that relate QA both with functionality and architectural styles. In addition, this process is fast to employ and easy to learn.

In the context of Altran, Quality Attributes are managed in a generic, non-systematic fashion. Even if they are somehow mentioned and described for most projects, they are never measured nor used to drive architectural design decisions. Therefore the type of methods, techniques and tools mentioned can be of great value to improve the way they deal with quality attributes. Since goal-oriented approaches don't seem to be very popular in industry, the use of Quality Attribute Scenarios may be more attractive and would benefit the company by introducing the notion of measure to each quality attribute. Additionally, the use of contribution tables, like suggested in Moreira's process [71], could raise the project team awareness when developing the functional architecture and the technical solution.

5.3. RQ3: What are the main techniques to analyze the risk of requirements?

Risk Management is a critical process of Project Management that aims at identifying, analyzing and mitigating potential risks of any business investment [2]. In particular, the risk analysis activity tries to answer the questions: *(i) What can happen? (ii) How likely is it to happen? (iii) Given that it occurs, what are the consequences?* [76].

Traditionally, in software development, Risk Management is performed after the design of the system to identify situations that can cause project failure [77]. However, most of the high level risks lie in the early phases of development and can affect the cost or schedule of the project [78]. Moreover, the introduction of risk mitigation strategies can cause requirements changes and force the revision of the initial design. It is well-known that fixing potential errors and preventing risks in the requirements phase is less costly than in later stages of development. Therefore, there is a need to perform Risk Management activities from the start of the project, integrating risk analysis within the Requirement Engineering process.

For that reason, also the CMMI model proposes a subpractice to perform risk assessment on the requirements (SP 3.4, subpractice 2: *"Perform a risk assessment on the requirements and definition of required functionality and quality attributes."*). This subpractice shows that the Requirements Development process area is indeed related to the Risk Management process area. Risk analysis for the requirements should then be incorporated in the generic Risk Management activities for the whole project.

At the requirements engineering level, risk analysis can be performed during requirements analysis to integrate risk mitigation countermeasures as part of the system's requirements. Risks are identified and analyzed along with the stakeholders' needs to elaborate countermeasures and to define risk-based criteria for selecting among alternative ways of fulfilling the requirements [77].

In the context of requirements analysis, a *risk* is considered any uncertain event that can keep a system from fulfilling its goals. These events are a set of uncertain circumstances, usually out of the control of actors, which can have an impact on the system (i.e., threats, failures and/or unintended happenings).

Risks are often measured with a combination of two attributes: the chance or probability of an event occurrence (*likelihood*) and the consequence of that event on the achievement of a goal (*severity*). When analyzing the risks, the aim is on reducing their likelihood, or mitigating the severity of its consequences.

5.3.1. Identification of Risks

Regarding risk identification, several techniques can be applied. According to [79], using a cross-functional team of developers and customers with different knowledge domains can be beneficial to the process because it sheds light on different kinds of risks (for example, technical, security, legal or environmental risks) and their correlation. The most common

techniques to identify risks are *brainstorming*, *workshops, questionnaires* and *scenario analysis*. A combination of such techniques is also possible. To aid in the identification process, inventories of common risks and lessons learned of similar projects can be used to seed the discussion and prevent repeating past mistakes. For that reason, the creation of a database of previous projects' risks grouped by application domain, might be part of a possible solution to satisfy both the Requirements Development and Risk Management process areas. Several generic software risks checklists can be found in the literature, some related with requirements activities ([80]–[82]). However, for the functional and non-functional requirements, their related risks are often specific to the application domain.

5.3.2. Analysis of Risks

Risk analysis techniques can be categorized as qualitative or quantitative. **Quantitative techniques** measures fix numeric values (such as the time or cost) of *severity* and *likelihood* of a risk. Conversely, **qualitative techniques** represent the *likelihood* and *severity* of a risk using an interval scale, where each interval has a label (such as low, medium and high) that includes a range of numerical values. Each technique type has its own strengths and weaknesses and can be used in combination.

Many current Risk Analysis methodologies found in the literature can only be used in later stages of the software life cycle. However, a few authors have recently attempted to perform it during earlier stages, creating frameworks that integrate existing risk assessment techniques in the requirements analysis.

5.3.2.1. KAOS framework

The KAOS framework [29] is a goal-oriented framework used for modeling requirements. However, it can also be used to model the risks associated to these requirements by using its concept of *obstacle* and *anti-goal*. An obstacle is a set of undesired behaviors that can lead to goal failure (unintended risks), while an anti-goal is a goal associated with malicious stakeholders (threats or intended risks). Although these features make the framework suitable for representing and analyzing requirements for secure and dependable systems, it does not provide a way of measuring the likelihood and severity of the identified risks.

5.3.2.2. Extension to KAOS

To bridge this gap, an extension to KAOS was proposed by Boness *et al.* [83] as a lightweight technique to assess the risks during requirements analysis. The idea is to build a KAOS goal-graph and annotate each goal with independent risks factors as illustrated in Table 5.6.

Table 5.6 - Risk factors, taken from [83]

Risk factor	***Description***
RF1:environment	Probability that the assumption can be satisfied by the environment
RF2:achievability	Probability that the requirement is achievable and can be satisfied by the software
RF3:refinement	Probability that the refinement is sound
RF4:mandate	Probability that the requirements are in line with stakeholders' needs

These risk factors quantitatively measure the probability of success. For example, if p is the probability of occurrence of a risk, then the metrics of Table 5.6 will measure 1-p. These probabilities are estimated with judgments made by the stakeholders or experts for some goals and propagated with simple calculations to obtain the remaining goals. In addition to the risk factors, each goal is also annotated with the percentage of its business *Value* and development *Cost*, in order to easily compare systems with different number of goals.

After all the goals are supplied with the proper annotations, a Risk Profile can be determined for the project. This is done in terms of the Feasibility and the Adequacy of the operacionalizable requirements. The Feasibility of requirements is the assertion that they are achievable and is measured by multiplying the risk factors RF1 and RF2. On the other hand, the Adequacy of

requirements ensures that what is set down is what is 'actually' wanted by the stakeholders and is measured using the risk factors RF3 and RF4. Having this two metrics calculated for each leaf goal, it is possible to draw plots of Feasibility versus Adequacy in terms of its Cost or Value, as shown in Figure 5.5. These plots place the operacionalized goals in 3 regions *("Proceed", "Proceed with caution", "Do not proceed*") that help managers to identify problematic goals associated to high-severity risks. The placement of regions is subjective and decided by the manager depending on the criticality of the project.

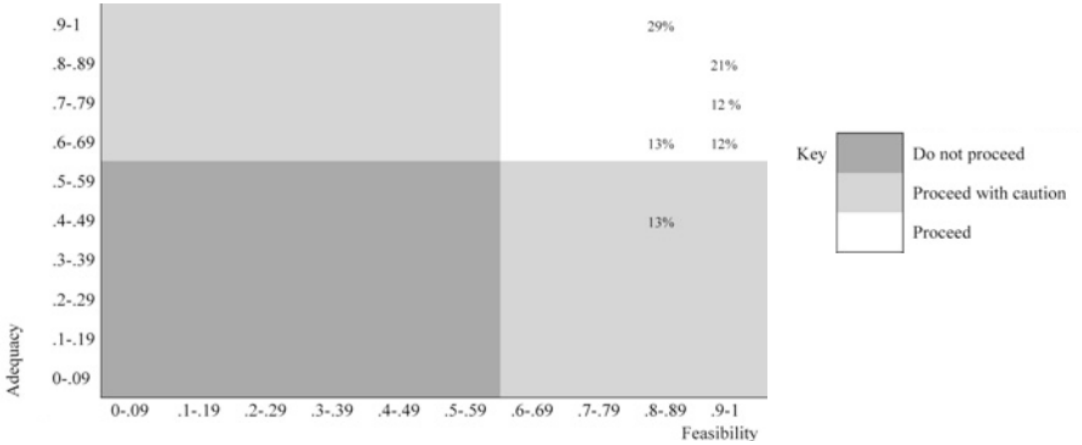


Figure 5.5 - Example of plot of leaf goal values (PValues) for Feasibility versus Adequacy, taken from [83]

This technique has been empirically tested with small and medium size projects, which showed that it is viable and not too time consuming for the experts [83]. In addition, it is also compatible with other non goal-oriented approaches or even with Agile projects. For instance, it is possible to derive the goal-graph from the conventional natural language specification or building smaller goal-graphs for each sprint. Nevertheless, this technique also suffers from some limitations. The fact that design alternatives (OR branches) are removed by producing additional alternative goal-graphs, could lead to a combinatorial explosion of graphs. Besides, it assumes that all leaf goals are independent and that obstacles are assessed by the goals that overcome them. To partially automate the technique, the authors built a tool called KAOS Lite [83].

5.3.2.3. Goal-Risk framework

Another goal-oriented approach for reasoning about risks during requirements analysis is the Goal-Risk framework, proposed by Asnar *et al.* [77]. This framework is based on the Tropos goal-modeling framework [84] and the DDP framework [85]. It is composed of three layers: assets, events and treatments. Figure 5.6 represents a goal-risk model divided in the three mentioned layers.

Assets are represented as goals that model strategic interests of stakeholders. In this layer, the goals are identified, decomposed and the dependencies among these goals are drawn. *Events* are uncertain circumstances that can be regarded both as risks and/or as opportunities, depending on their negative or positive impact on the assets. An event is qualitatively characterized by the *likelihood* of its occurrence (*Likely, Occasional, Rare and Unlikely*) and by the *severity* of its effects (depicted as the sign of the impact of an event on a goal ++, +, -, --). *Treatments* are tasks that aim to mitigate the risks either by reducing its *likelihood* or by attenuating its *severity*. Each of these artifacts has the attributes SAT(*n*) and DEN(*n*) that represent available evidence of satisfaction/denial of the artifact *n*, respectively. Their values are measured qualitatively using the scale *(F)ull, (P)artial* and *(N)one* evidence. The three layers of the framework are related through four types of dependencies that can be *AND/OR decompositions* to refine the artifacts of each layer, positive or negative *contributions* among the artifacts, positive or negative *impact relations* of events on assets and *alleviation relations* that connects treatments with a negative impact relation (a risk) to attenuate its severity.

To analyze and evaluate the alternative sets of solutions introduced by OR decompositions in the model, the authors propose four algorithms and a graphical tool (GR-Tool) that automates them. This framework was validated with projects from multiple domains and the results show that it is rather fast to learn and use in early phases [77]. It is the only framework that deals both with risks and opportunities and takes into account the dependencies among artifacts. Also, it is flexible as it allows the extension of features of other Risk Analysis frameworks (FTA, FMECA, Markov-model) and can be used to assess risk with multiple actors. On the down side, this framework uses the notion of SAT/DEN evidence, which is not as clear as probability, and the qualitative evaluation doesn't assess precisely the risk.

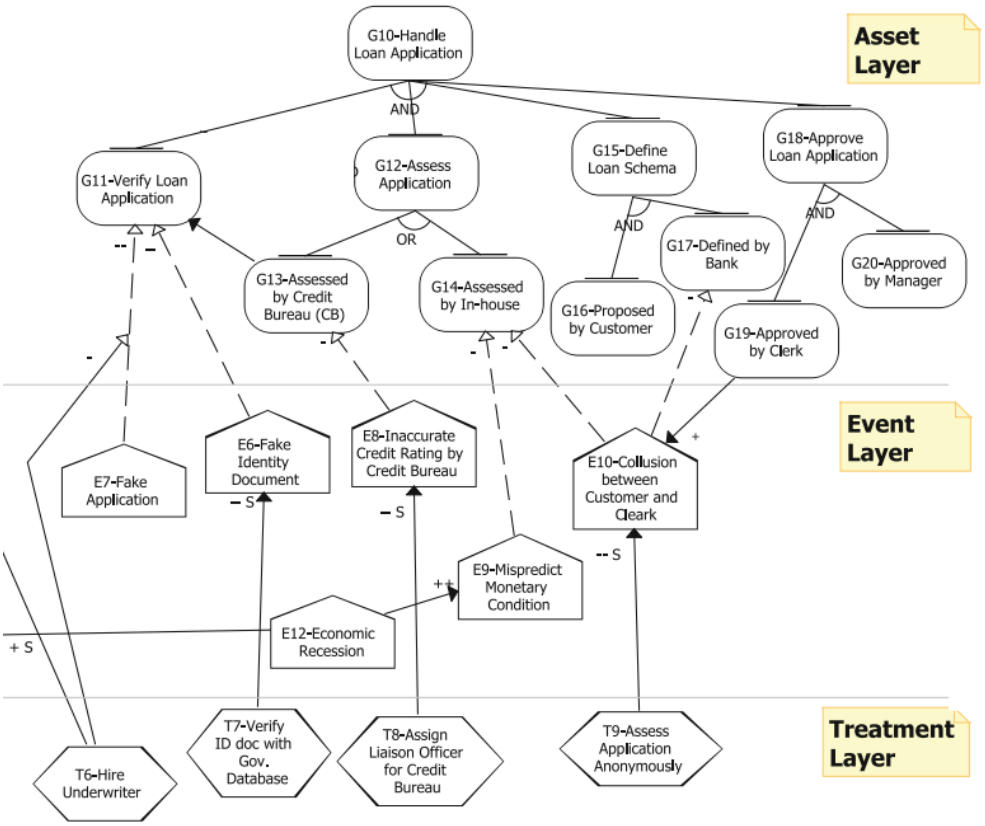


Figure 5.6 - Example of a Goal-Risk Model for a Loan Application System, adapted from [77]

5.3.2.4. Framework based on UML

Amber *et al.* [78] developed a framework based on UML to identify risky functional requirements during requirements analysis. This framework transforms functional requirements into UML scenarios with associated risks as alternative scenarios and then transforms scenarios into control flow graphs. This help to find the McCabe's cyclomatic complexity of a scenario based specification (number of edges – number of nodes + 2). The scenarios are also converted into an UML's sequence diagrams so that the number of message at a failure mode is obtained. The failure is also subjectively assigned with a severity *s* (with 0<s<1) and the risk factor *Rf* for that scenario can be calculated with the formulas (1) and (2).

$$Rf = \text{complexity} * \text{severity} \quad (1)$$

$$\text{complexity} = \text{McCabe's cyclomatic complexity} * \text{Number of message} \quad (2)$$

To conclude, possible mitigation countermeasures are identified and selected for the high risk scenarios. The authors don't refer to any tool to automate the process neither to the validation of the framework on pilot projects.

5.3.2.5. CORAS

CORAS [86] is a UML-based framework for risk analysis of security-critical systems, particularly focused on aspects of IT security. It integrates aspects from multiple risk analysis techniques (such as HazOp, FMEA, FTA) modeled and documented using a graphical UML-based language to represent threats and risks. The CORAS risk management process is conducted in five steps: context identification, risk identification, risk analysis, risk evaluation and risk treatment. To support this framework, there is an open-source tool (the CORAS tool), which facilitates the communication by non-experts using simple UML based diagrams and allows the reuse of risk analysis results. Figure 5.7 shows an example of a threat diagram created with the CORAS tool for a telemedicine project.

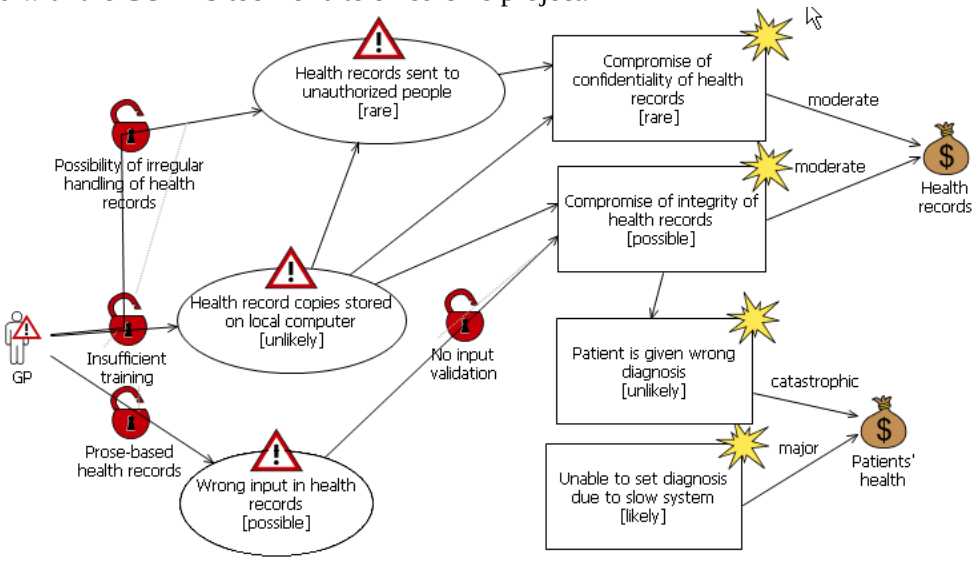


Figure 5.7 - Example of threat diagram created with the CORAS Tool, taken from [87]

This threat diagram is one of the four kinds of diagrams available with the CORAS tool. It could be complemented with assets diagrams, risk diagrams and treatment diagrams. All these diagrams share a common subset of graphical symbols. Once the risks, their likelihood and consequences are identified, they should be placed into a risk evaluation matrix to extract the ones that need further analysis. Although this framework is mostly used to model risks at an enterprise level, it can also be used to model the risk during the requirements analysis.

5.3.3. Risk Management at Altran

Altran Portugal has a defined a Risk Management process that is applied at project level, as shown in **Figure 5.8**. It is often performed by the Project Manager, who acts as Risk Manager, during the planning phase, along with the elaboration of the Project Management Plan. During the remaining development phases, there are also weekly meetings to track the project's progress, in which the risks are also identified with the involvement of the project team.

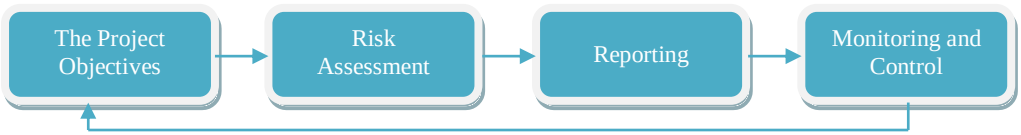


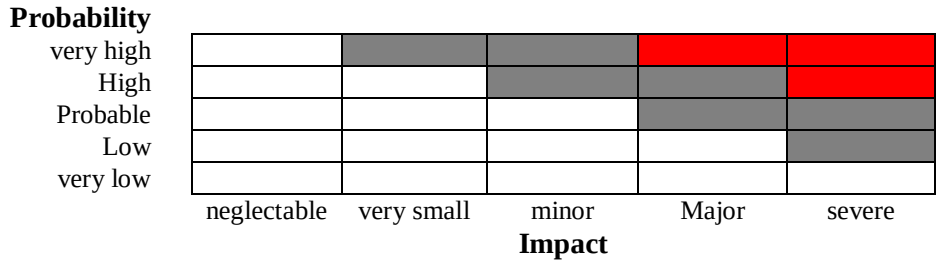
Figure 5.8 - Risk Management Process

Risk Assessment includes Risk identification, Risk Analysis and the definition of a Risk Response Plan. In the **Risk Identification** activity, the project team identifies potential problems through the analysis of the deliverables and expert judgment input from relevant stakeholders. The identified risks are recorded into a Risk Register spreadsheet that is stored along with the remaining project documentation in the Knowledge Tree.

Following the identification of risks, **Risk Analysis** is performed both quantitatively and qualitatively. For the qualitative analysis, the *probability* of each risk is evaluated as 'very

high', 'high', 'probable', 'low' and 'very low'. The *impact* of a risk is assessed with respect to the other risks as 'neglectable', 'very small', 'minor', 'major' and 'severe'. The impact and probability can also be measured quantitatively using a scale from 1 to 5. Risk's *impact* and *probability* are then mapped into a Risk severity Grid (see Table 5.7), so that the priority of each risk can be assessed and countermeasures elaborated for the high level risks. Three priority levels can be assigned to a risk, depending on the region of the grid: 'Risk Management mandatory' (red area), 'Risk Management recommended' (grey area) and 'Risk monitoring only' (white area).

Table 5.7 - Altran's Risk Severy Grid



A **Risk Response Plan** is then elaborated for the risks of the mandatory and recommended priority groups. It will identify strategies that minimize the effects of risks, and that can be classified according to the following types: 'Avoidance', 'Transference', 'Sharing', 'Mitigation' and 'Acceptance'.

5.3.4. Discussion

During the analysis of this research question, we concluded that most of the studies found follow a goal-oriented approach. They model and reason about requirements risks, aiming at minimizing them while fulfilling the top-level goals.

Although the **KAOS framework** [29] is able to represent risks as obstacles, it does not consider their likelihood or severity. To bridge this gap, the **KAOS Extension** [83] allows to measure quantitatively the probability of success of each goal, along with an estimation of its value and cost. This technique is rather useful as it helps managers to identify problematic goals associated to high level risks. However, it does not consider dependencies among goals and can be problematic due to the explosion of the number of goal-graphs. Using a similar approach, the **Goal-Risk framework** [77] is able to measure the likelihood and severity associated with each risk. Conversely to the previous technique, the measurement is qualitative and takes into account dependencies among goals. Nonetheless, Goal-Risk framework requires acquaintance with complex algorithms and uses a notion of "Satisfaction/Denial" to measure artifacts, which is harder to understand than the notion of probability. Contrarily to goal-oriented approaches, the **UML-based framework** [78] proposes the use of known artifacts like scenarios, workflows and sequence diagrams to identify and analyze risks. It helps managers to identify high risk scenarios through its complexity and severity. However, the proposed technique doesn't consider the value or cost of each risk, doesn't provide an automated tool and doesn't show evidences of any validation in real projects. As another UML-based approach, the **CORAS** method [86] uses a set of diagrams that share a simple graphical notation. As the previous techniques, it considers the probability and impact of risks. However, it is usually applied to model only aspects of IT security.

To incorporate the risk assessment of requirements in the current Risk Management process used at Altran, the existing notion of *probability* and *impact* can equally be applied to requirements risks. However, similarly to the risk analysis techniques found in the literature, the business value and cost of implementation of the affected requirement should also be taken into account since two risks with the same probability and impact can have very different meanings, depending on how important the requirement is to the customer or on the number of modules it will affect. In addition, the responsibility for the monitoring and resolution of each identified

risk should be clearly specified. It cannot be solely associated with the Risk Manager as in the current process.

Therefore, a new Requirements Risk Register spreadsheet needs to be created, incorporating in the existing table the responsibility for the risk, the requirements it will affect and their associated value and/or cost. It is important to note the notion of risk and opportunity of a certain event: it can have a negative impact on one requirement, but help to fulfill other. As suggested in [77], [83], the creation of a goal graph with high level goals can help the team to visualize how a risk and its countermeasure affects the other requirements.

To support the identification of requirements related risks, the requirements documents should be analyzed, possibly along with the prioritization of requirements. In particular, special attention should be given to the non-functional requirements as they usually come with a few common risks (for example a hacker breaking in a system can be a risk associated with security). In fact, due to their nature, quality attributes impose restrictions that when not met can themselves be seen as project risks. Also, the analysis of the alternative cases of the established scenarios could help uncover risks of functional requirements (for example system failures).

5.4. RQ4: How to define scenarios and operational concepts?

The definition of scenarios is a mature RE technique that is often used during elicitation, analysis and validation of requirements. In CMMI, the usage of scenarios and operational concepts appears as a recommended practice under the analysis and validation of requirements.

According to CMMI, "*a scenario is a sequence of events that may occur in the development, use, or sustainment of the product, which is used to make explicit some of the functional or quality attribute needs of the stakeholders. In contrast, an operational concept for a product is a general description of the way in which an entity is used or operates and usually depends on both the design solution and the scenario*." In other words, an operational concept can originate distinct scenarios. For instances, an operational concept for an activity that can be performed either manually or automatically can derive two distinct scenarios.
In the analysis and validation context, Sutcliffe [88] defined scenarios as "*facts describing an existing system and its environment including the behavior of agents and sufficient context information to allow discovery and validation of system requirements*". In other words, they describe the interactions between user roles and the system in a defined scope (such as a single system, a department, the entire organization) and time-frame (such as an operation, a day, the all system life).

Scenarios are applicable to all domains and can be used throughout the whole system life-cycle, including operations, installation, development, maintenance, support or disposal. Typically, they are used to describe functional goals. This can be done either informally through a narrative scenario or more formally using, for example, sequence diagrams. A complementary approach that combines both formal and informal descriptions is also a common practice. Alexander and Maiden [89] identified six types of scenarios:

- ***Story*:** a narrated description of a connected sequence of events in some domain, or more usually of actions taken by a small number of interacting protagonists. In Agile development, brief *User stories* are a common practice.
- ***Situation, Alternative World*:** a projected future situation or snapshot. They are particularly useful for long-term business or government planning.
- ***Simulation*:** models static or dynamic situations and aspects of scenarios that have a direct concrete interpretation. It can "give precise answers about whether such a scenario could be realized with any plausible design" or "to evaluate the implications of alternative possible worlds or situations".
- ***Storyboard*:** a sketch, or a sequence of drawings, used to describe a user interface or to tell a story. Mainly used in Human–Computer interaction to define the user interface screens.

- ***Sequence*:** a list of (possibly numbered) interactive steps taken by independently acting agents playing roles. They include Operational Scenarios, Concepts of Operations and Test Cases.
- ***Structure*:** an imposed structure to describe the content of a scenario. It includes narrative descriptions or diagrammatic representations like flowcharts, UML Activity or Sequence diagrams, Negative Scenarios and Misuse Cases and the particularly popular Use cases.

Each of these scenario types is a better fit to different phases of the life cycle and to different roles of people writing and reading the scenario [89]. Still, stories and sequence of events are perceived as the most useful as they are generally understood universally without any training.

Use Cases are the most popular approach used in both industry and academia to define functional scenarios. Several templates have been proposed to write the content of use cases. Most of them adopt the fields of the proposed template in Table 5.8.

Table 5.8 - Use Case Description Example

Title	Get Status on Order
Description	The customer consults the status of a placed order
Main Actor	Customer
Secondary Actors	None
Pre-conditions	A valid user has logged into the system
Main Scenario	1. The use case starts when the customer tries to consult the status of an order 2. The customer fills in the data with the identifier of the order, the customer identifier or the customer name 3. If the customer entered the identifier of the order a) The system shows an order 4. Otherwise, if the customer entered the customer's identified a) The system returns a list of all orders of that customer b) The customer selects one from the list of orders c) The system displays the selected order
Post-conditions	None
Extension Scenario 1	At step 3, if the order code does not match actual orders a) The system displays a message informing inexistence of order
Extension Scenario 2	At step 4, if the customer identifier does not match actual customers a) The system displays a message informing inexistence of customer

For Cockburn [90], a well written use-case should follow a more detailed structure composed of: *title, primary actor, goal in context, scope, level, stakeholders and interests, precondition, minimal guarantees, success guarantees, trigger, main success scenario, extensions, technology & data variations and related information*. However, he also admits a more simplified structure that can be used when fewer details are needed. Either way, the title of a scenario should be carefully selected using an active-verb that represents the overall goal of the actor; the main scenario should contain 3 to 9 steps and the Extension scenarios should describe other things (such as failures) that can happen, and must be handled.

There is often a misconception between use cases and use case diagrams. While the first textually describes a list of interactive steps between actors and a system, the later merely summarizes the various ways that people and systems may interact in an oversimplified diagram; it does not contain an actual scenario.

According to [91], besides the connection between use cases and use case diagram, a relationship between Use Cases and Quality Attribute Scenarios can also be established.

As shown in Figure 5.9, the actors are the source of the *stimulus*, the prerequisites would be described in the environment and the steps which represent both the trigger, i.e. the stimulus of a scenario, and the response or the outcome of that stimulus. Nevertheless, this correlation is not necessarily one-to-one. It assumes that the related use case and QA scenarios are describing the same information. Therefore, it is possible to have scenarios with no use cases and vice-versa.

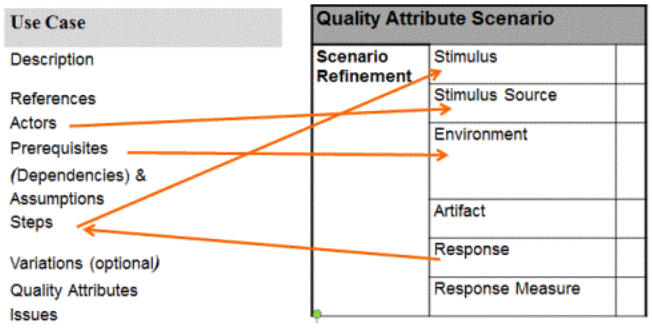


Figure 5.9 - Relationship between Use Cases and Quality Attribute Scenarios, taken from [91]

There are several simple tools that can be used to define scenarios, like text editors or word processors with template support. For large and complex system requirements, dedicated use case tools like *CaseComplete, RequisitePro* or *Wiki software* might also be helpful. Most UML Tools like *MagicDraw* or *StarUML* also support both narrative scenarios and visual modeling of use cases, activity or sequence diagrams.

5.4.1. Scenario definition at Altran

To understand how this practice is currently performed at Altran, the requirements related templates and the documentation of five distinct projects was examined (*LESS*, *Anacom*, *AltranREQ*, *WSpace* and *ISINOV* projects). The existing Requirements Specification templates do not include a template for Use Cases.

The first evident problem found during the analysis of projects is that there is not a common approach to define scenarios. Not all requirements documentation had descriptions of scenarios. Three of them adopted the same Use Case template, as they were led by the same Project Manager, and one described scenarios in term of processes through activity diagrams. The three projects that included Use Cases were also supported with Use Case diagrams.

An example of a typical Use Case defined in those projects is given in Table 5.9. Through this example it is possible to understand that there is an incorrect use of the field '*Description*', '*Main Scenario'* and *'Alternative Scenario'* according to the proposed template of Table 5.8. Also, analyzing the described steps, one can notice that the interaction is mostly one-sided: responses given by the system are not always mentioned or not declared at all.

Table 5.9 - Typical Use Case defined in Atran's Projects

USE CASE 1.	**Remove Project**
Actor	Administrator.
Description	1) The user selects the desired project from the list, or alternatively use(s) the field(s) search: a) The administrator can search for a Project Manager, Functional Analyst, Project Name; 2) The user selects the option to remove the project; 3) The user confirms the removal of the project.
Evaluation Criteria	Check that the project no longer exists in the list of projects.
Pre-Condition	Realization of USE CASE User Login.
Main Scenario	The confirmation message of the removal of the project is displayed.
Alternative scenario 1	The message that there are requirements associated with the project is displayed.
Post-Condition	The project is removed from the system.

One of the analyzed projects described scenarios through activity diagrams with swimlanes. This is particularly useful as it helps to visualize the actors, decisions and order of steps in a process. However, they do not solve the problem of identifying all the if-then-else branches, in particular with exceptional events that need to be handled. According to Alexander ad Maiden [89], Use Cases are a better fit to this problem as they provide more information and context. Using both Use Cases and activity diagrams allows to generate and trace deterministic straightforward Test Cases by finding the set of "paths" through each branched Use Case.

5.4.2. Discussion

The use of scenarios is a powerful approach to analyze functionality in engineering [89]. It allows simplifying complex system's interactions at the lowest possible cost by describing them as plain-text stories. This bridges the communication gap between customers and developers as they are understandable by both technical and non-technical people. Although scenarios are insufficient as system specifications, they are very valuable during requirements discovery. Additionally, as scenarios carefully describe sequences of interactions performed by intelligent agents, they can also serve as basis to develop functional tests, acceptance tests and can even be used to validate requirements through walkthroughs. Nevertheless, ensuring the consistency of multiple stories and avoiding ambiguities is still an evident limitation of this approach inherent to the use of natural language. Selecting the proper granularity for the scenario's description or integrating them with other requirements engineering approaches is still subject of discussion and research in academia.

The methodology currently used at Altran could be enhanced by adding this Use Case description template to the general Requirements Specification Template and making it a mandatory field for all projects. Also, providing an example might help to clarify the current misinterpretation of the fields. The usage of well-specified Use Cases can help to generate and trace black-box test cases, using them for scenario-testing. In these circumstances, the derivation of activity diagrams from Use Cases could be an advantage, as they can help to visualize the set of possible paths for the functional Test Cases. Therefore, our proposal is that both use cases and activity diagrams should be specified during the functional analysis. According to the CMMI guidelines, these scenarios should also be established not just for operations as it is currently practiced, but also for installation, development, maintenance, support and disposal. Since their definition is an iterative process, we recommend them to be included in the peer reviews, to be refined and ensure their consistency with the requirements.

5.5. Summary

Throughout this chapter, the solutions to bridge the main gaps found were studied. The results of each research question were discussed and presented to Altran. The solutions selected are summarized in Table 5.10.

Table 5.10 - Summary of Solutions for each Research Question defined

Research Question	***Solution for Altran***
Q1 *What are the main techniques used to prioritize requirements?*	Adopting easy to use techniques that take little time to perform, like the MoSCoW or Simple Raking techniques, are the most suitable for Altran given its simplicity. Prioritization should be applied at the customer requirements level, based on the customer's perceived importance.
Q2 *How to derive the architecture using Quality Attributes?*	Goal-oriented techniques are not very common in industry. Therefore, using simpler techniques, like Quality Attribute Scenarios, would benefit the company by introducing the notion of measure to each attribute. Also, the use of contribution tables for trade-offs analysis can raise the project team awareness when developing the functional architecture and the technical solution.
Q3 *What are the main techniques to analyze the risk of requirements?*	Incorporating the risk assessment of requirements in the current Risk Management process. Create a new requirements risks spreadsheet evaluating, besides the impact and probability, the business value and cost of such risk, the requirements affected, possible mitigation countermeasures and the responsible for monitoring the risk. Risks identification performed along with the prioritization, with special attention given to Quality Attributes.
Q4 *How to define scenarios and operational concepts?*	Adding a Use Case template to the general Requirements Specification document, providing a complete example to avoid misinterpretations of the fields. Also, activity diagrams are suggested, to support the functional analysis and the generation of functional tests.

6. New Requirements Management and Development process

The CMMI Gap Analysis conducted in Section 4.4 revealed that the current requirements process used at Altran needs some improvements. In this chapter, the proposed changes to the current process will be described in Section 6.1, along with the refinements made as the process matured in Section 6.2, and an impact evaluation of these changes in the remaining processes of the SGD, in Section 6.3. In addition, a new CMMI assessment of the proposed requirements process will be conducted to demonstrate the improvements in compliance, in Section 6.4.

6.1. Proposed Methodology

The process proposed in this dissertation is based on the current Requirements Management process with a few changes in the methodology used. These suggested modifications may concern the terminology, the workflow's activities or the templates currently used at Altran.

6.1.1. Terminology Definition

The current Requirements Management process does not provide a clear definition of the terminology used in its description. Therefore, this terminology needs to be defined and used consistently throughout the process, hence improving its coherence. The proposed concepts and their definitions were based on the CMMI terminology [2] and adapted to Altran's context. The glossary created is presented in Table 6.1.

Table 6.1 - Glossary of the proposed terms

Definition	Description
Stakeholder	A group or individual that is affected by or is in some way accountable for the outcome of an undertaking. Stakeholders may include project or work group members, suppliers, customers, end users, and others [2].
Customer	The party responsible for accepting the product or for authorizing payment. Customers are a subset of stakeholders [2].
Stakeholder's need	The raw information obtained directly from each stakeholder about the problem to solve.
Customer Requirement	The result of eliciting, consolidating, and resolving conflicts among the needs, expectations, constraints, and interfaces of the product's relevant stakeholders in a way that is acceptable to the customer [2]. They are user-oriented and represent the problem to solve.
Solution Requirement	A refinement of the Customer Requirements into the developers' language, making implicit requirements into explicit derived requirements. The developer uses Solution Requirements to guide the design and building of the product or service [2].
Component	A deployable, independent unit of software that is completely defined and accessed through a set of interfaces [1]. It is generally a lower-level component of the product and is integrated with other components to "build" the system [2].
Functional Requirement	A requirement that specifies a function that a system or system component must be able to perform [3].
Interface Requirement	A requirement that specifies an external item with which a system or system component must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction [3].
Quality Attribute Requirement	A property of a product or service by which its quality will be judged by relevant stakeholders. They are characterizable by some appropriate measure. Quality attributes are non-functional, such as timeliness,

	throughput, responsiveness, security, modifiability, reliability, and usability. They have a significant influence on the architecture [2].
Technical Requirement	A requirement that refers to the technical aspects that the system must fulfill and that limit the solution space beyond what is necessary for meeting the given functional and quality attribute requirements. For instances restrictions on the technology or the hardware used.

As we can see on the glossary table, requirements are considered in Customer and Solution abstraction levels that denote the difference between *user-oriented* and *developer-oriented* requirements. This distinction is proposed to evidence the clear separation between Customer and Product Requirements required by the CMMI. The term 'Customer Requirement' was adopted directly from CMMI, however, in this context, 'customer' is actually intended to include other relevant stakeholders. We chose to not adopt the term 'Product Requirement' from the CMMI terminology due to possible ambiguity issues that will be explained in Chapter 7.

The concepts defined in the glossary of Table 6.1 are, of course, interconnected. These relationships are illustrated in the model depicted in Figure 6.1.

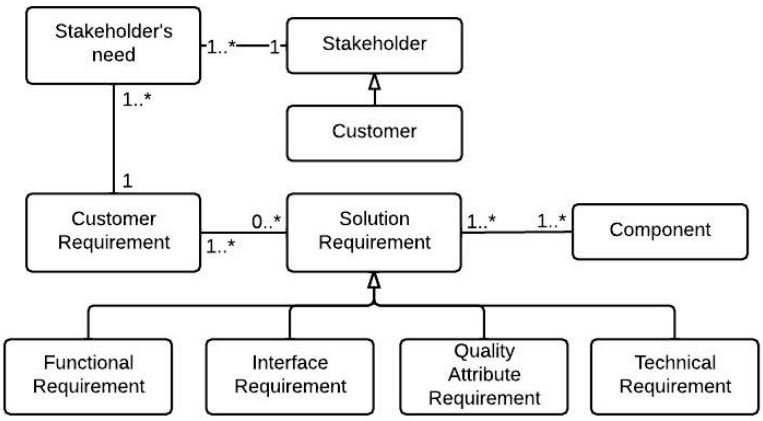


Figure 6.1 - Model of domain concepts

As shown in this figure, the stakeholders, including the customers, provide information about their needs. These needs are then formally written as Customer Requirements, which can gather several needs in a single requirement. A Customer Requirement is then detailed into zero or more Solution Requirements to acknowledge those situations that it is decided to not addressed the Customer Requirement. The Solutions requirements are usually a specification of a single Customer Requirement. However, due to the reuse practices proposed in the next Section, we considered a one-to-many relationship for those reusable Solutions Requirements that can satisfy several Customer Requirements. Solution Requirements can be of the functional, interface, quality attribute or technical type and are grouped by their logical components.

6.1.2. Workflow Modifications

Regarding the process workflow, the modifications proposed may concern the addition of new activities or the update of existing ones. This improved workflow is illustrated in Figure 6.2, in which the new activities are represented in red, updated names are in bold and previously existing activities are represented in white.

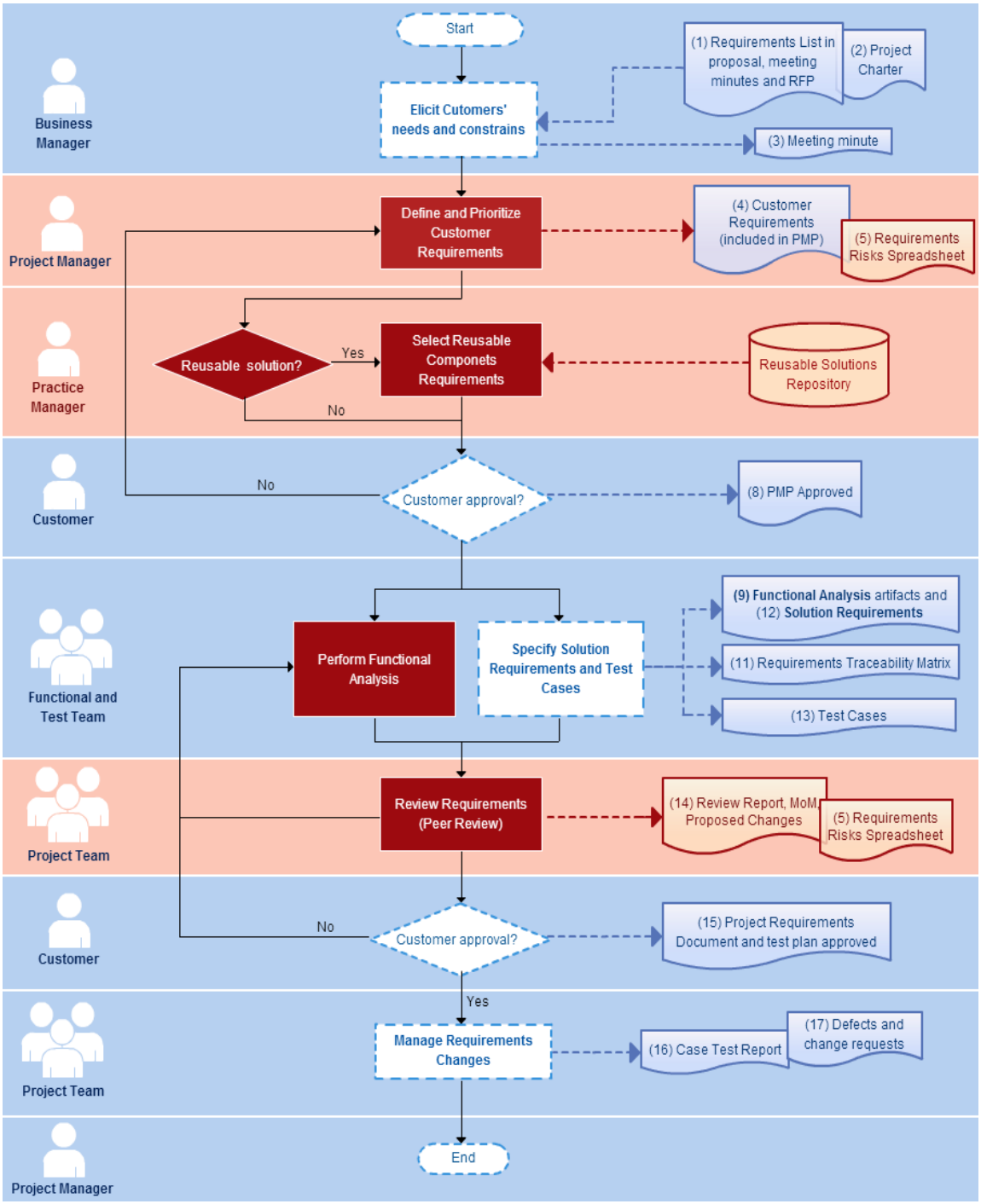


Figure 6.2 - Workflow of the new process

The list of the inputs and outputs of each activity is defined in Table 6.2. Each activity will then be described systematically, with a description, inputs, outputs and a table listing the involved roles and their responsibilities.

Table 6.2 - Inputs and Outputs of the proposed requirements process activities

Activity	Inputs	Outputs
Elicit Customers' needs and constraints	(1),(2)	(3)
Prioritize Customer Requirements	(1),(2),(3)	(4),(5)
Reusable solution?	(1),(3),(4)	(6)
Select Reusable Component Requirements	(6)	(7)

Customer approval?	(4),(5),(7)	(8)
Perform Functional Analysis	(4),(7)	(9)(10)(11)
Specify Solution Requirements and Test Cases	(4),(10)	(11)(12)(13)
Review Requirements (Peer Review)	(9),(11),(12),(13)	(14), (5)
Customer approval?	(9),(11),(12),(13)	(15)
Manage Requirements' Changes	(11),(13),(15)	(16),(17),(5)

(1) Proposal, RFP; (2) Project Charter; (3) Meeting minutes; (4) Customer Requirements; (5) Requirements Risks; (6) Reusable Solution or Components; (7) Reusable Solution or Components' Requirements; (8) Approved PMP; (9) Functional Analysis; (10) Derived Requirements; (11) Traceability Matrix; (12) Solution Requirements; (13) Test Cases; (14) Peer Review Report; (15) Project Requirements Approved; (16) Test Cases' Report; (17) Defects and change Requests

The **Elicit Customer's Needs and Constraints** activity is actually the unchanged 'Customer Requirements' activity in the current workflow. As in the current process, it is the responsibility of the Business and Practice managers and aims at understanding the problem to be solved by identifying general needs, expectations and constraints provided by the customer during the project proposal. We chose to simply update this activity's name to explicit its intent.

Inputs: The project proposal, requirements identified by the customer, RFPs, meeting minutes or other sources, formal or informal, of information.

Outputs: Meeting minutes containing the list of customer's needs, expectations and constraints.

Table 6.3 - Roles and Responsibilities of Elicit Customer's Needs and Constraints activity

Role	Responsibilities
Business Manager and Practice Manager	Collect information about the customer's needs, constraints and expectations. Handover information to the Project Manager.
Project Manager	Accept assignment, and understand customer's provided information.

The **Define and Prioritize Customer Requirements** activity partially matches the 'Analysis Requirements List' activity of the current process with a few additional tasks. It aims at translating all the stakeholders' needs and constraints gathered into a prioritized set of Customer Requirements. As in the previous activity, these Customer Requirements will be analyzed by the project team and integrated in the Project Management Plan. It is suggested (but not mandatory) the use of *User Stories* to represent these Customer Requirements, as it is considered a simple technique that manages to capture the needs and constraints in the users' perspective with the right level of abstraction. Regarding the prioritization, we suggest the use of easy methods like a simple Ordinal Scale Ranking in which the *N* Customer Requirements are assigned a priority of 1 to *N*, or according to the MoSCoW method in which each requirement is assigned a priority from the set {MUST, SHOULD, COULD, WON'T}. During this activity, the risks related to requirements should also be assessed and documented in the Requirements Risks Register as described by Atran's Risk Management process.

Inputs: Meeting minutes with the gathered stakeholders' needs and constraints, a list of requirements identified by the customer, the project proposal.

Outputs: Prioritized list of Customer Requirements documented as User Stories in the PMP and the Project Requirements Specification Document; the Requirements Risks Spreadsheet.

Table 6.4 - Roles and Responsibilities of the Prioritize Customer Requirements activity

Role	Responsibilities
Project Manager	Choose the Prioritization Method to be used. Validate the Customer Requirements and its assigned priorities. Validate the risks related to requirements.

Project Team	Create the User Stories. Assign Priorities. Assess the risks related to requirements.

The **Reusable Solution** gate and **Select Reusable Component Requirements** activity aim at identifying the existence of similar, reusable solutions or components developed previously that can satisfy the Customer Requirements. The goal is to allow the reuse of Solution Requirements knowledge by extracting the specific component requirements that have been written in previous projects. This will reduce the requirements analysis effort, test plan effort, save project time, increase compliance with internal standards, and increase reliability by using previously validated requirements. Given that the Practice Managers are responsible for the project's initial solutions, they should be the roles with most experience and knowledge about solutions developed in the past. Hence, the Practice Manager is the role proposed as the responsible for these reusability activities. Moreover, we propose the creation of a Reusable Solution's Repository that could be consulted by Project Managers, so that this reusability could be independent of the Practice Manager expert judgment. Further details on such repository are given in Section 9.3.1. The development of a *procurement process* to guide this reusability is also suggested, so that the practices proposed can evolve from opportunistic to planned reuse.

Inputs: The Customer Requirements.

Outputs: The reusable Solution Component Requirements documentation.

Table 6.5 - Roles and Responsibilities of the Reusable Solution gate and Select Reusable Component Requirements activity

Role	Responsibilities
Practice Manager	Verify and decide if previous solutions or components exist that can meet the Customer Requirements. Inform and deliver the reusable Solution Component Requirements documentation to the project team. Insert reusable solutions or components in the repository.
Project Team	Insert or reference the reusable information in the Project Requirements Specification document.

The **Customer Approval** gate is unchanged from the current process. It aims at presenting the Project Management Plan to the Customer and getting his formal approval. In this Project Management Plan, the prioritized Customer Requirements should be included along with information about possible requirements reuse.

Inputs: The PMP with the Customer Requirements Prioritized and its associated risks.

Outputs: Customer approved PMP and, therefore, approved Customer Requirements.

Table 6.6 - Roles and Responsibilities of the Customer Approval decision gate

Role	Responsibilities
Customer	Review and approve the Customer Requirements and its priorities assigned (included in the PMP)
Project Manager	Present the PMP and get the customer approval.

The **Perform Functional Analysis** activity is performed in parallel with the specification of Solution Requirements and Test Cases. It aims at analyzing the behavior of the system, describing what the system is intended to do, in order to support the Solution Requirements development and test cases. Several techniques are proposed to perform functional analysis, such as Use Cases, behavior diagrams (e.g., activity, state, sequence diagrams), quality attribute scenarios, quality attribute trade-off tables, component diagrams and graphical interface

mockups. Further details on these produced artifacts are described in the Project Requirements Specification Template (see Section 6.1.3). This functional analysis was already performed previously as part of the Requirements specification activity. However, given its importance in the development of requirements, we decided to clearly state it in the workflow and systematically describe it as a separate activity.

Inputs: The Customer Requirements, the Solution Requirements (that are being built simultaneously), the PMP and the meeting minutes with the stakeholders.

Outputs: Functional Analysis chapter in the Project Requirements Specification document containing the Functional Description with scenario specification, the Functional Architecture and the User Interface mockups; Derived Solution Requirements.

Table 6.7 - Roles and Responsibilities of the Perform Functional Analysis activity

Role	Responsibilities
Functional Consultants	Create the Functional Analysis chapter in Project Requirements document with Scenarios, Functional Architecture and User Interfaces.
Project Manager	Validate created artifacts and guide the Project Team.

The **Specify Solution Requirements and Test Cases** activity matches the 'Requirements and test case specification' activity of the current process. It should be performed in parallel with the Perform Functional Analysis activity and aims at refining the approved Customer Requirements into detailed Solution Requirements specifications for the development team to implement into the project work product. These Solution Requirements should concern the product functionality as well as product quality attributes, their interfaces and technological constraints. These requirements should be grouped in the document by their logical components to facilitate modularity and future reuse.

Inputs: Customer Requirements, Project Management Plan

Outputs: Solution Requirements, Test Cases, Traceability Matrix

Table 6.8 - Roles and Responsibilities of the Specify Solution Requirements and Test Cases activity

Role	Responsibilities
Project Manager	Validate the solutions requirements, test cases and traceability matrix
Functional and Test Team	Create the Solution Requirement chapter in the Project Requirements document (exported from Testlink) Create the test cases for each requirement Create the traceability of requirements

The **Review Requirements (Peer Review)** activity aims at formally verifying the produced requirements documentation to ensure that it is clearly specified, correct, necessary and sufficient. This internal validation should be performed in a meeting with the Project Manager acting as the moderator, the authors of the reviewed documents and other external reviewers. These external reviewers can be a part of the project team, like programmers or testers, and are an important presence in the meeting to get an independent perspective over the produced requirements. The main goal is to identify defects and ambiguities, checking if the set of requirements actually define the system to develop, if they are understandable and do not show inconsistencies with others and if they are viable (if what they specify can be implemented). All the defects found should be recorded in a log by one of the external reviewers.

Inputs: The Project Requirements Document, Requirements Risk Register and the Requirements traceability matrix.

Outputs: The Requirements Peer Review Report signed by the team and the Requirements Defects found.

Table 6.9 - Roles and Responsibilities of the Review Requirements (Peer Review) activity

Role	Responsibilities
Project Manager	Promotes the peer review meeting Nominates and notify reviewers
Project Team	Identifies defects

The second **Customer Approval** gate is kept from the current requirements management process. However, in addition to the formal customer's approval of the requirements specification documents, actual validation of the requirements should also be considered. For that matter, the presentation of prototypes is suggested to get the customer's feedback. This can be as simple as interactive GUI Mockups or more complex simulation prototypes, depending on the criticality of the project or on what was previously agreed with the customer.

Inputs: The Project Requirements Specification Document, in particular the GUI mockups acting as prototype, the Traceability Matrix and the Test Cases.

Outputs: The Project Requirements Specification and Test cases acceptance term signed by the customer and a meeting minute with the provided feedback.

Table 6.10 - Roles and Responsibilities of the second Customer Approval decision gate

Role	Responsibilities
Customer	Give feedback on the presented product representations. Approve the Project Requirements Specification and Test Cases.
Project Manager	Present the product representations to the customer. Present /Deliver the Project Requirements document and Test Cases. Promote and collect formal approval.

The **Manage Requirements Changes** activity replaces the "Development", "Test Execution" and "Work Product approved?" activities of the current process. In fact, we didn't consider it as a new activity since all of its tasks were already somehow performed previously. This decision was based in the fact that the Development and Test phases are not a part of the requirements phase. Therefore, they should not be mentioned in a requirements process. Instead, they should be placed as activities of the Execution process. The Manage Requirements' Changes activity that replaced the previous ones takes place during these phases. The reasons behind such requirements' changes are often related to the resolution of defects detected when running a test case; an uncompleted set of requirements; functionality changes requested by a customer; modifications or expansions of a requirement specification during the design or development. When such a change is necessary, the impact that it will have on other requirements should be analyzed by the Project Team, using the requirements traceability matrix. If such a change affects the Plan or the scope of the project, than the Project Manager should deploy a change request, depending on the severity or extent of the actions needed to address the change.

Inputs: Project Requirements, Traceability Matrix, Test Cases

Outputs: Test Cases' Report, Defects, Change Requests, Updated Risks

Table 6.11 - Roles and Responsibilities of the Manage Requirements Changes activity

Role	Responsibilities
Project Manager	Deploy change requests.

Project Team	Analyze the impact of the change on the requirements and the plan. Update Requirements Documentation. Update Requirements Risks.

The map presented in Figure 6.3 summarizes the activities of the new process and their respective deliverables.

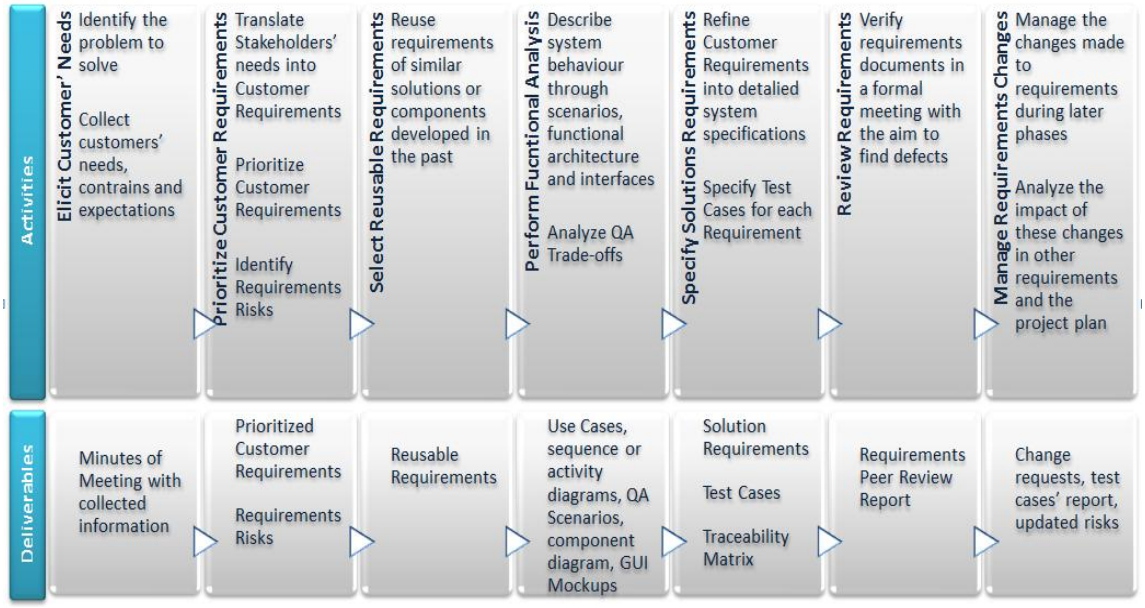


Figure 6.3 - Summary of activities and respective deliverables

6.1.3. Templates Modifications

The current requirements management process used at Altran provides two templates for Requirements specification and analysis (*Feature Requirements Specification* and *Functional Design Specification*). To improve consistency, it was agreed with Altran that only one document should be created. Even if it will considerable increase the volume of information present in a single document, it will also avoid having several scattered documents that depend on each other, which will aid maintaining traceability. Therefore, the two existing templates were merged and updated to create a single **Project Requirements Specification template** containing the following separate chapters:

- Customer Requirements
- Product Requirements
- Functional Analysis

The mind map of the new improved template is represented in **Erro! A origem da referência não foi encontrada.** with the new sections added to the template illustrated in red. The result is a well-structured and improved Requirements Template that is fully compliant with CMMI RD best practices. Each of the 3 major sections corresponds to an output of a different activity in the workflow and addresses a different goal of CMMI RD.

The **Customer Requirements** chapter is an output of the "Define and Prioritize Customer Requirements" activity. In this chapter the relevant stakeholders' roles are briefly described and their needs, expectations and constraints are formally documented. The format of User Stories is suggested, but not mandatory. This will help to understand the several viewpoints of the project and resolve possible conflicts. A User Story template and examples of well written user stories are given in the document. This set of user stories is then given a priority, assigned with the MoSCoW method or with a simple ordinal ranking, as defined in the process.

The artifacts produced by this chapter are: Stakeholders' description and Prioritized User Stories.

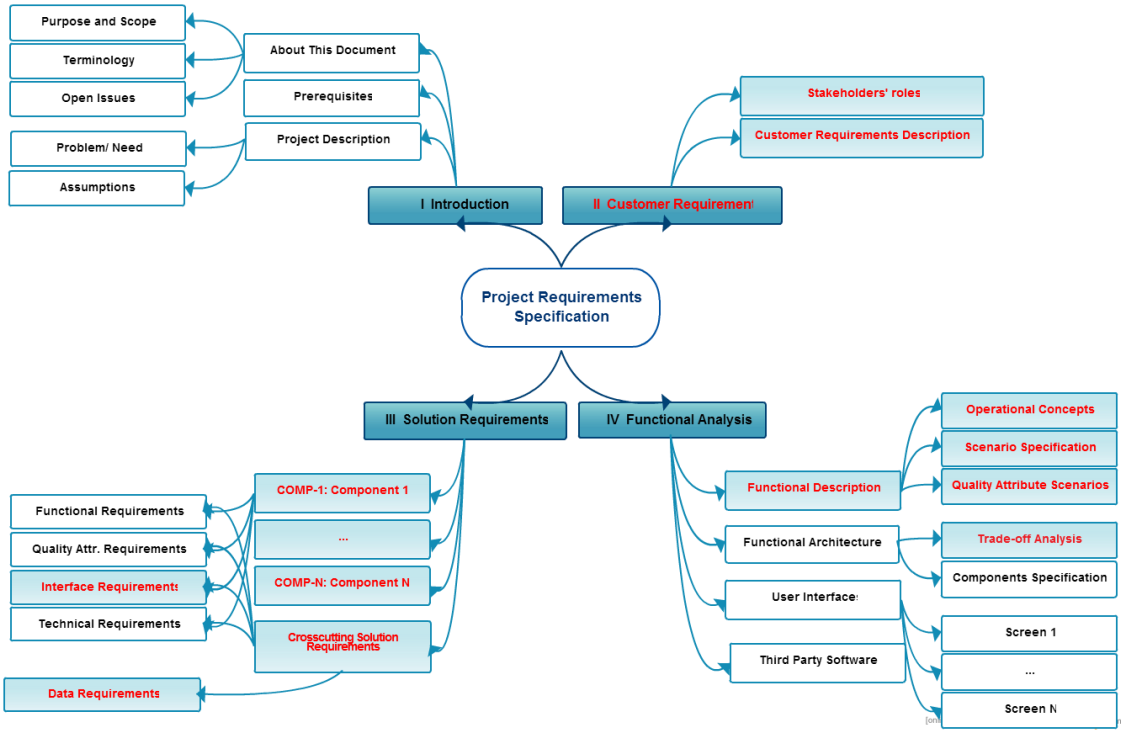


Figure 6.4 - Mindmap of the new template created

The **Functional Analysis** chapter corresponds to the output of the "Perform Functional Analysis" activity. This chapter was adapted from the existing *Functional Design Specification template*. In this chapter, the behavior of the product is analyzed with the specification of scenarios, the functional architecture and the definition of User Interfaces. To specify Scenarios, a Use Case template is provided and Activity or Sequence diagrams should be created to support the Use Case descriptions. Quality attribute specific scenarios are also a suggested. Regarding the Functional Architecture, a quality attribute trade-off analysis is recommended. Table templates for this analysis were created. A component diagram, with brief descriptions of each component should also be created. The User Interface chapter previously existent was adapted to include mockup sketches of the Graphical User interfaces.

The artifacts produced by this chapter are: Use Cases, Activity or Sequence diagrams, Quality Attribute Scenarios, Quality Attribute trade-off tables, Component Diagram, GUI Mockups.

The **Solution Requirements** chapter is an output of the "Specify Solution Requirements and Test Cases" activity. This chapter was adapted from the existing *Feature Requirements template*. The Customer Requirements are detailed into specifications according to the previous existing template. The major change in this chapter is the organization of these requirements by its logical components. This will facilitate modularization and future reuse of component requirements. A new section regarding interface requirements was also created. Because some requirements are transversal to the application, a "Crosscutting Solution requirements" section is provided to place those requirements that are common to several components. Finally, a data requirements section is proposed, in which a domain model should be created with the respective entities' specification.

The artifacts produced by this chapter are: Functional Requirements, Quality Attribute Requirements, Interface Requirements, Technical Requirements, Entity Domain Model, specification of entities' fields.

To guide the project team in completing this requirements document, examples of the expected artifacts are provided throught the various sections. Additionally, to cope with the possible

inexperience of some consultants with quality attributes, a **Quality Attributes Catalog** containing quality attribute definitions, Quality Attribute Scenarios examples and a general Quality Attribute Contribution matrix is provided. The aim of this supporting document is to raise awereness about the importance of quality attributes, how to deal with the most common and the impact they can have on arquitectural decisions.

Besides the Project Requirements Template, another template was created to support the "Peer Review" activity. The **Requirements Peer Review Report template** should be used as a checklist by the review team during the review meeting, and aims at verifying the requirements documentation generated (Project Requirements Specification and Traceability Matrix) regarding its format, content and traceability of the created artifacts. The goal is to ensure that the produced requirements are clear, complete, consistent and correct. Each section provides items with checkboxes and a Defect log where the identified defects should be recorded. Each defect found should be briefly described and characterized with one or more of the following types:

- **Omission (O):** Necessary information about the system has been omitted from the requirements document (e.g. the interface to an external entity is not specified).
- **Ambiguous Information (A):** Information within the requirements document is inconsistent with other information in the document or ambiguous, i.e. any number of interpretations may be derived that should not be the prerogative of the designer.
- **Incorrect fact (I):** Some sentence contained in the requirements document asserts a fact that cannot be true under the condition specified in the general requirements or general domain knowledge.
- **Extraneous (E):** Information is provided that is not needed or used.
- **Miscellaneous (M**): Other defects.

Furthermore, the person responsible to solve the defect found should be documented, along with the deadline date to solve it.

6.1.4. Other Modifications

In addition to the changes made to the workflow and to the templates, other general improvements concerning the overall process were considered. Examples of such improvements include updates in traceability, tools and the roles' responsibilities.

6.1.4.1. Traceability

As explained in Section 4.2, Requirements Traceability is already addressed at Altran. In particular, traceability was documented for the requirements dependencies, the connections between requirements and test cases and, when possible, the connections between requirements and source code. This notion of traceability had to be improved to accommodate the new division between Customer and Solution Requirements. Furthermore, traceability between Solution Requirements and Functional Analysis artefacts like Use Cases, activity diagrams, Quality Attributes Scenarios and GUI mock-ups should also be maintained. This is especially important to support the reusability practices proposed. A model of the new traceable items and their relationships is shown in Figure 6.5.

The traceability between Customer Requirements, Solution Requirements, Components and Test Cases can be achieved using the TestLink tool. However, for the remaining artifacts, this traceability is only achieved through the documentation, as the IDs of the artifacts (see Table 6.12) are associated to the respective Solution Requirements. Furthermore, the items represented with dashed lines are not considered traceable by the proposed methodology. The Stakeholders' needs are gathered in minutes of meeting or in the project proposal and, therefore, it is not possible to trace them to the Customer Requirements. On the other hand, although the traceability between Solution Requirements and Lines of Code is suggested in the previous process, it is hard to achieve and is actually not being performed for most projects. For completeness purposed, user stories are also represented in this Figure. However, in reality, they are the format used to represent Customer Requirements.

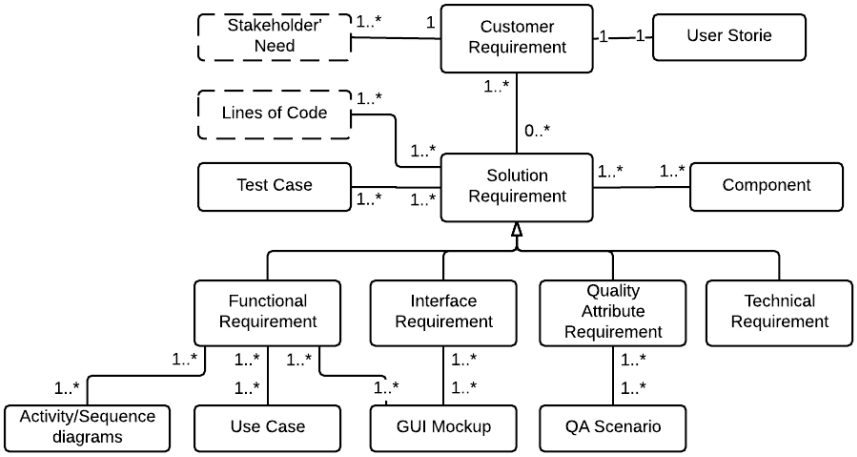


Figure 6.5 - Traceability between artifacts

Each artifact should have an unique ID, with the format defined as in Table 6.12, where SEQ is a natural number representing the sequence number of the artifact and TYPE is a value from {F, I, QA, T}, representing the functional, quality attribute, interface or technical type of the respective requirement.

Table 6.12 - IDs of the created artifacts

Item	ID
Customer Requirement	CR <SEQ> - <TYPE>
Solution Requirement	SR <SEQ> - <TYPE>
Component	COMP <SEQ>
Use Case	UC <SEQ>
QA Scenario	QAS <SEQ>
GUI Mockup	SCREEN <SEQ>
Test Case	TC <SEQ>

6.1.4.2. Tools

The set of tools used in the current process was maintained (see Section 4.2.3). Testlink will still be used to document all the requirements and test cases. Its ability to structure knowledge in a tree of folders will allow maintaining the separation between Customer and Solution Requirements. Likewise, splitting Solution Requirements by its components is also possible using a folder as a component.

Nevertheless, two new tools were proposed to the set of tools used until now. To create all the diagrams needed to develop and analyse requirements, *MS Visio* was the natural choice as it was already used by most project teams. Similarly, *Balsamiq Mockups* was suggested to create sketches of the Graphical User Interfaces necessary for the Requirements Functional Analysis and that can be used as storyboards for validation with the clients.

6.1.4.3. Roles and Responsibilities

The roles established previously at Altran, and listed in Section 4.2.1, were preserved. However, the responsibilities were extended according to the new activities in the REQM&D process.

In summary, the **major changes** to the current process were:

- Separation of Customer and Solution Requirements
- Prioritization of Customer Requirements
- Systematic Functional Analysis
- Emphasis on Quality Attributes and their impact on architecture
- Peer Review activity
- Reusability of Requirements

6.2. Methodology Adjustments

The new Requirements Management and Development process presented in this chapter evolved and matured during the writing of this dissertation, but particularly during the validation of the process. This validation included a meeting with the experts to gather their feedback, which led to some process adjustments to better reflect the Project Managers' needs at Altran. These changes made to our original process and artifacts proposal are documented next. The major changes impacted on the terminology, process workflow and documentation templates. Further details on this validation meeting held on 2014.08.06 with Altran's Project Managers are discussed in Chapter 7.

6.2.1. Changes to the Terminology

Regarding the proposed terminology, the Business Requirements concept was removed from the original glossary table (see Table 6.1). This type of requirement was suggested to represent the business' top level goals that we felt should be included in the Project Management Plan.

Table 6.13 - Business Requirement definition

Definition	Description
Business Requirements	The high level requirements or top goals of the project that will be included in the Project Management Plan.

However, these requirements descriptions were excluded from the process because, as some Project Managers pointed out, using such high level requirements in the PMP would increase the difficulty of the effort estimations performed by the Practice Managers. Besides, having a third requirements abstraction level was considered of irrelevant value to the process.

Additionally, the term "*Solution Requirement*" evolved from the CMMI original term "*Product Requirement*", which was not well accepted, as Altran is not a product house. The term "Software Requirement" was also considered. However, it could narrow down the vocabulary to software solutions, which might not always be the case.

6.2.2. Changes to the Workflow

The workflow presented in this chapter also undergone some changes relatively to its initial version. Our original process workflow proposed is illustrated in Figure 6.6.

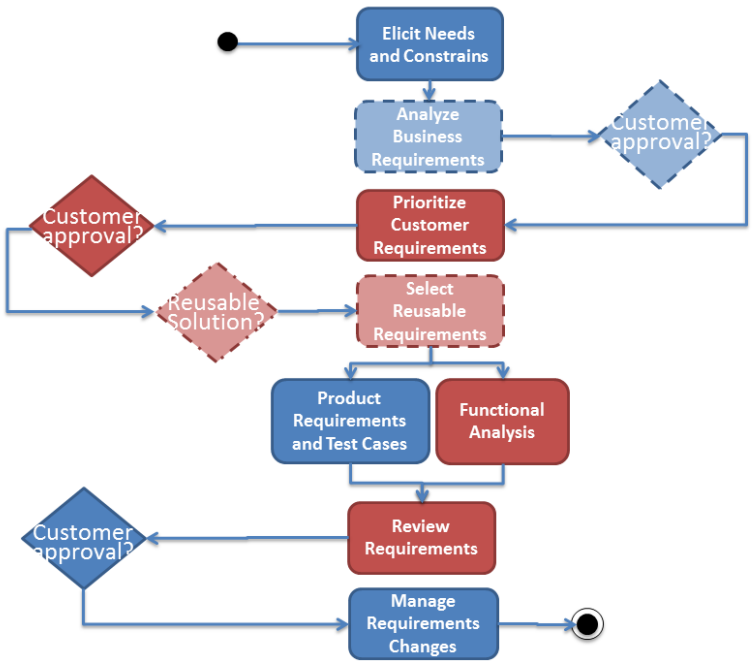


Figure 6.6 - Initial version of the REQM&D Workflow

In this first version, there was an "Analyze Business Requirements" activity in which the business requirements were identified and included in the Project Management Plan. As the business requirements were removed from the terminology, we decided to move the customer requirements prioritization to earlier in the process, including these prioritized customer requirements both in the Project Management Plan and the Project Requirements Specification document. This would also save one approval moment with the customer.

Likewise, the requirements reuse activities were shifted to before the kickoff presentation, so that this possible reuse information could be also included in the Project Management Plan.

The "Perform Functional Analysis" and "Specify Solution Requirements" parallel activities were switched due to the fact that functional consultants usually start by building scenarios.

6.2.3. Changes to the Documentation Templates

The major modifications in the templates were due to the changes in the terminology used or in the workflow's activities. In particular, chapters III and IV (Functional Analysis and Solution Requirements) were switched to reflect the shift in the respective workflow activities.

6.3. Impact on the other SGD Processes

As mentioned in Section 4.3, the current requirements process is closely related and can be integrated with two other processes in the SGD[3]: the Project Management process and the Execution process. Therefore, the impact that these proposed modifications may have on such processes should be analyzed.

6.3.1. Project Management process

The project management process activity flow was not affected by the process changes. However, the descriptions of its activities and decision gates should be updated to comply with the new requirements terminology defined.

In addition to this minor updates, the **Risk Management process**, presented as an appendix of the project management process, should also be updated to accommodate the risks related to requirements. The risk process, its roles and responsibilities are unchanged. However, the *Customer and Solution Requirements should be included as artifacts to review* in the risk identification phase. In particular, regarding the risks documentation and reporting, a *new Risk Register spreadsheet specific for recording requirements risks should be created*. This spreadsheet is similar to the current Project Risk Register as it includes an analysis of probability of occurrence versus impact on the project in case it does. However, it also needs to include the Business Value for the customer, the Cost of implementing its mitigation, and a defined responsible to monitor it.

6.3.2. Execution process

The Execution process must be subject to a deeper analysis, in particular, regarding the *Prepare* and *Design* phases.

In the **Preparation phase** we decided to remove the "Identify Customer Requirements" activity since we considered it to be a part of the requirements process. Instead, the description of the "Give inputs to PMP" activity was extended and updated to reflect all the information involved in the PMP, including inputs from the proposed requirements process, such as the prioritized customer requirements, requirements risks and possible reusable requirements.

Regarding the **Design phase**, its "Requirements Specification" activity should be renamed and updated to encapsulate the "Perform Functional Analysis" and "Specify Solution Requirements and Test Cases" parallel activities of the new Requirements Management and Development

[3] Sistema de Gestão de *Delivery*

workflow. This activity was renamed to "Develop Project Requirements" and its description updated to explain its intent and reference the Requirements Management and Development process. In addition, there is a one-to-one correspondence between the first "Peer Review" activity and the "Review Requirements" activity of the REQM&D process.

This correspondence between the several processes activities should be stated in the processes' description. To get a clear understanding of how the several processes are interrelated, the workflow of Altran's macro processes is illustrated in Figure 6.7. As shown in this Figure, the Prepare phase encapsulates the Customer Requirements prioritization and the Requirements reusability activities. Likewise, the Design phase includes the Functional Analysis, the specification of Solution Requirements and the Peer Review activities.

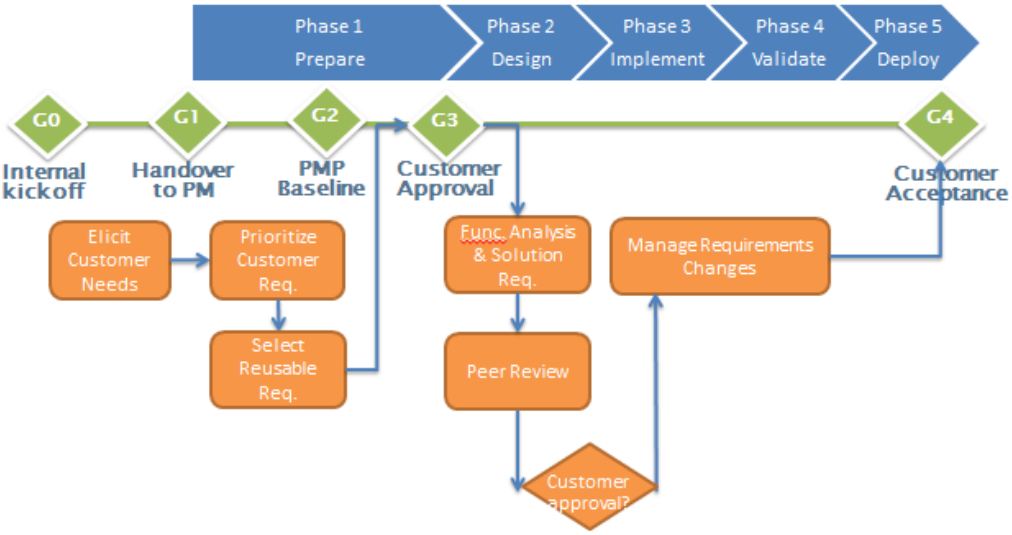


Figure 6.7 - Macro workflow of Altran's Processes

6.4. Gap Analysis of the new REQM&D process

In this section, the proposed methodology was subject to a new CMMI assessment, using the same evaluation method defined in Section 4.4.1. This assessment was conducted based on the new process description and the new templates created to demonstrate their compliance with the CMMI Requirements Development process area practices. As in the previous evaluation, each of the RD specific practices (see Table 2.3) will have their subpractices rated and justified accordingly.

The full PII table for Requirements Development process area, providing all the new subpractices' ratings and objective evidence supporting it, is available in Appendix D.

The elicitation of needs of **Practice SP 1.1** is performed in the "*Elicit Customer's needs and constraints*" activity by the Business and Practice Managers, whom are free to choose the elicitation technique to use. This is usually done through Interviews and brainstorming with the customer during the development of the proposal. If needed, the project team can also meet with different stakeholders during other activities of the requirements process to solve possible ambiguities. These needs, expectations and constraints elicited are recorded in meeting minutes.

Practice SP 1.2 is performed in the "*Define and Prioritize Customers Requirements*" activity, when the information gathered from stakeholders is formally documented in the Project Requirements document and the PMP, possibly in the format of user stories (subpractice 1). These user stories are then prioritized through the *MoSCoW* method or with a simple ordinal Ranking, regarding the importance of each user story to the customer (subpractice 2). The constraints of subpractice 3 are stated under the assumptions and constraints of the PMP.

Practice SP 2.1 is performed in the "*Specify Solution Requirements and Test Cases*" activity. The Customer Requirements are analyzed and detailed into technical terms in the Project Requirements Specification document. This includes the architectural requirements mentioned in subpractice 3, in particular quality attributes and technical requirements. Associated to the Quality Attributes are quality attribute scenarios, in which a measure is defined. Subpractice 2 is also met as the functional analysis activity, performed in parallel to the Solution Requirements specification, might derive new requirements that are documented similarly. Both the solution and Customer Requirements have a proper ID and are stored in Testlink so a traceability matrix can be generated.

Practice SP 2.2 is performed in the "*Specify Solution Requirements and Test Cases*" activity. The developed Solution Requirements are organized in the document according to section headings representing the logical components, so subpractice 2 and 3 are met. During the "Define and Prioritize Customer Requirements" activity, the requirements can be divided into work packages and work packages into deliverables; therefore subpractice 4 is also met. As mentioned previously, traceability is kept for Solution Requirements and components in the traceability matrix (subpractice 5).

The interface requirements of **Practice SP 2.3** are also established in the "*Specify Solution Requirements and Test Cases*" activity as, for each component, a section for Interface Requirements is provided in the document. To support this practice, a component diagram is used to explicit internal interfaces and their interactions. For external GUI interfaces, mockups are created which can derive new interface requirements that should be added to the Solution Requirements Specification chapter.

Practice SP 3.1 is performed in the "*Functional Analysis*" activity, in which scenarios and operational concepts are specified in the Functional Analysis chapter of the Project Requirements template (subpractice 1). A Use Case template is provided to systematically define scenarios. Later in the process, in the Peer Review activity, the specified scenarios will be reviewed, satisfying subpractice 3. The definition of the environment (subpractice 2) is performed in the technical design specification document.

Practice SP 3.2 is performed during the "*Functional Analysis*" activity. Each identified scenario should be analyzed and associated to an activity or sequence diagram. The Activity diagrams can be supported with the description of each activity input, output and actors. Regarding the Quality Attributes, they are described with a template in the Solution Requirements chapter and as quality attribute scenarios in the Functional Analysis chapter. Therefore subpractices 2 and 4 are met. Besides, the trade-off analysis included in the functional architecture section reflects the influence that Quality Attributes have in architectural design decisions (subpractice 3). As mentioned before, subpractices 5, 6 and 8 are accomplished since the requirements are grouped according to their logical components. Practice 7 is partially met as Customer Requirements are not directly allocated, however it is possible through traceability. The key business drivers of subpractice 1 are identified by the Business Manager during the proposal.

Practice SP 3.3 is performed in the "*Peer Review*" activity. During this review, the Project Requirements Specification and the Traceability Matrix will be carefully analyzed by all team members to ensure correctness, consistency and completeness (subpractice 3). This includes the analysis of scenarios and operational concepts (subpractice 6), as well as verifying if Solution Requirements satisfy the Customer Requirements (subpractice 2). Subpractice 1 is met since Customer Requirements are mapped to their functional components. The technical measures mentioned by subpractice 5 are specified in the quality attribute scenarios defined in the documentation.

Practice SP 3.4 is performed throughout the process. The Customer Requirements risks are initially accessed during the "*Customer Requirements prioritization*" activity (subpractice 2). The interface prototypes created during "*Functional Analysis*" are analyzed to balance the needs and constraints provided by the stakeholders (subpractice 1). The quality attribute trade-off analysis is revised during the peer review (subpractice 4) and the risks related to all functional

and non-functional requirements are re-accessed (subpractice 2 and 3). To help discover the risks the use cases and QA scenarios can be analyzed and a list of similar projects lessons learned consulted. A Requirements Risks spreadsheet should be used to record the all the risks found, their monitors and actions to mitigate them.

Practice SP 3.5 is performed in the last "*Customer Approval*" decision activity. Requirements Validation with the customer includes the presentation of prototypes that can be simple interface mockups or other more sophisticated prototypes depending on the customer and criticality of the project (subpractice 2). Subpractice 3 is partially met as the maturity of the design is assessed in the peer review and during project development, as the PM is responsible to validate all the artifacts created by the team. However, possible validation issues identified are not analyzed or documented. Subpractice 1 is partially met by the Requirements Risk Management in the Peer Review activity.

The measurement of the subpractices allows estimating the score of their respective practices by the percentage of subpractices that were met by Altran's processes. The result is shown in Table 6.14.

Table 6.14 – Assessment results of the proposed process

GOALS	PRACTICIES	Rating
SG1 Develop Customer Requirements	SP 1.1 Elicit Needs	100%
	SP 1.2 Transform Stakeholder Needs into Customer Requirements	100%
SG2 Develop Product Requirements	SP 2.1 Establish Product and Product Component Requirements	100%
	SP 2.2 Allocate Product Component Requirements	90%
	SP 2.3 Identify Interface Requirements	100%
SG3 Analyze and Validate Requirements	SP 3.1 Establish Operational Concepts and Scenarios	88%
	SP 3.2 Establish a Definition of Required Functionality and QA	81%
	SP 3.3 Analyze Requirements	100%
	SP 3.4 Analyze Requirements to Achieve Balance	75%
	SP 3.5 Validate Requirements	67%

This new gap analysis showed that the current process is indeed compliant with the specific practices of CMMI Requirements development. The overall evaluation results from the current and proposed processes were gathered in Table 6.15 to demonstrate its evolution.

Table 6.15 – Assessment results relative to both current and proposed processes

GOALS	PRACTICIES	RATING current	RATING proposed
SG1 Develop Customer Requirements	SP 1.1 Elicit Needs	100%	100%
	SP 1.2 Transform Stakeholder Needs into Customer Requirements	67%	100%
SG2 Develop Product Requirements	SP 2.1 Establish Product and Product Component Requirements	63%	100%
	SP 2.2 Allocate Product Component Requirements	80%	90%
	SP 2.3 Identify Interface Requirements	75%	100%
SG3 Analyze and Validate Requirements	SP 3.1 Establish Operational Concepts and Scenarios	12%	88%
	SP 3.2 Establish a Definition of Required Functionality and QA	75%	81%
	SP 3.3 Analyze Requirements	50%	100%
	SP 3.4 Analyze Requirements to Achieve Balance	25%	75%
	SP 3.5 Validate Requirements	33%	67%

As Table 6.15 reveals, when comparing the specific practices' scores of the former process with the improved one, it is visible that the overall process is more compliant with CMMI and the

major improvements affected the practices 3.1, 3.3 and 3.4. In fact, 5 out of 10 practices are now 100% performed and the remaining 5 are largely implemented, which indicates a very positive impact in requirements quality and a better control of its documentation.

6.5. Summary

Throughout this chapter, the new Requirements Management and Development process proposed for Altran was described. In our proposal, we intended to adapt the current process, establishing a new terminology, new activities and updating its templates, without harming the requirements management practices formerly achieved.

The major modifications concern the separation of requirements in Customer and Solution levels, the prioritization of Customer Requirements, the systematic functional analysis with an emphasis given on Quality Attributes, the review of requirements in a formal peer review meeting and the opportunity for reusability activities.

Moreover, the impact that such modifications might have in other related SGD processes was studied, revealing that only minor adjustments in the other processes were needed.

To demonstrate that the new process meets the non-compliances found earlier, the same SCAMPI-based evaluation method defined earlier was applied to the developed process. Comparing the both assessment results, we can conclude that the compliance level has indeed increased, with all the recommended practices evaluated as fully or largely implemented.

7. Process Validation

The new Requirements Management and Development process proposed in this dissertation was validated and tested to ensure its feasibility and adequacy to the company. Such validation was conducted in three different moments, using different approaches to minimize the chances of validity threats: (1) the process was presented and discussed with three project managers and one practice manager and their feedback was collected; (2) the process was also applied to a case study to illustrate the templates application; (3) the process was submitted to an independent "live-test" in a small pilot project to assess its suitability. The feedback collected was used to improve the overall process, terminology and templates. Ideally, the process could be further validated and tested in real-life projects in order to ensure its feasibility and adequacy to the company. This is, however, out of the scope of this dissertation. The remaining of this Chapter discusses with some detail those three validation moments in Sections 7.1, 7.2 and 7.3, respectively. To conclude, Section 7.4 provides the lessons learned during the validation activities and Section 7.5 summarizes this Chapter.

7.1. Project Managers and Consultants Feedback

Validating the proposed process with experienced consultants and Project Managers that will actually have to apply it in real projects is an important stage when implementing a new process in a company. Gathering their feedback will help to assess the process acceptance and understand their major concerns, increasing the chances of future adherence. To evaluate the acceptance of the process proposed in this dissertation, a meeting with three Project Managers and one Practice Manager was conducted during two and a half hours. Prior to this meeting an online questionnaire (Appendix E) and a 20 min presentation with the new process, emphasizing the changes compared to what existed were prepared. During this meeting the experts' opinions were discussed and collected to produce a new version of the process, and the questionnaire was made available at the end of the meeting.

In general, the reactions to the proposed process were positive, but with some, expected, resistance to change. All the reviewers recognized that the new methodology would increase the quality of the requirements and therefore, the quality of the final product. According to the survey, the process was perceived as "*clear*" and "*complete*". However, it was also considered "*idealistic*". During the meeting some concerns were collected regarding weather all the activities and artifacts to produce could be managed in the short time given for most projects. Therefore, it was suggested that some sections of the project requirements template, like the "*quality attribute trade-off analysis*", should not be mandatory for all projects. Other changes suggested are described next.

7.1.1. Terminology

Regarding the proposed terminology, the definition of terms and the use of different levels of abstraction for requirements were well accepted. Still, the term "Product Requirements" adopted from the CMMI terminology led to some misinterpretations. Some reviewers claimed that Altran, as a consultancy company, focuses on projects instead of products. Adopting this term directly from CMMI could cause ambiguities as they are not a "product company". The term "Solution Requirements" was suggested and later adopted.

7.1.2. Business Requirements

The use of Business Requirements to include in the Project Management Plan was also pointed out as potentially problematic. Using such high level requirements concept would difficult the cost estimations performed by the Practice Managers during this phase. Therefore, it was proposed the exclusion of Business requirements from the glossary and terminology, using, instead, Customer Requirements for estimation and inclusion in the PMP, which would also reduce the number of activities in the process.

7.1.3. Project Requirements Specification Template

The fusion of the existing two requirements templates in a single document may facilitate consistency, but it might have also its disadvantages. It was pointed out that using a single document might be a problem for some clients due to security and confidentiality constraints. According to one Project Manager, some requirements documents have to be split in several for approval by different stakeholders that have different access permissions to classified information. Therefore, as shown in the survey, everyone agreed that a single document should be kept, but with the possibility of splitting it in smaller documents when necessary.

7.1.4. Customer Requirements Prioritization

The survey confirms that the prioritization methods were well accepted by all the reviewers. However, it was discussed whether using *User Stories* to describe Customer Requirements could be potentially difficulty to some consultants that are not familiar with the technique. Still, 67% of the respondents agree that it would be a suitable technique.

7.1.5. Techniques

It was suggested that some techniques or notations exemplified in the project requirements template should not be mandatory. For instance, the component diagram in the Functional Analysis chapter could be either modeled in UML, as proposed, or with other notation chosen by the Project Manager. The level of awareness of each technique proposed is shown in Figure 7.1. The results indicate that training in the proposed techniques is fundamental, given that not all the respondents seem to be experienced with them.

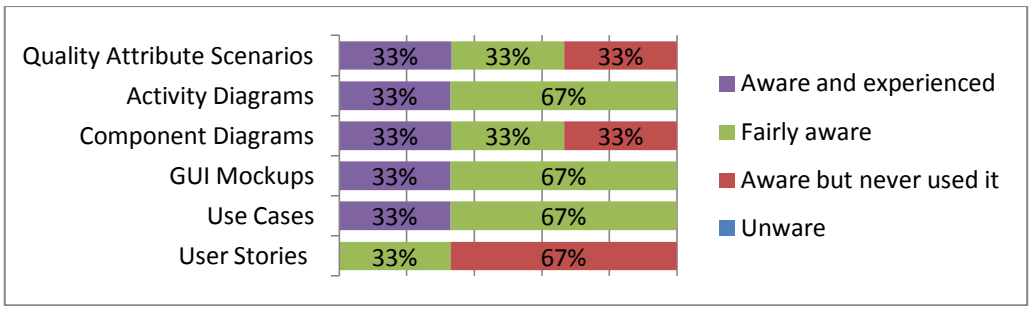


Figure 7.1 - Knowledge of the techniques proposed

7.1.6. Requirements Reuse

The requirements reuse practices were accepted by half of the reviewers (see Figure 7.2). The other half considered them to be indifferent because as consultancy company, their projects are very diverse, adapted to the customer and therefore, hard to reuse. Moreover, it was suggested that they could be performed earlier in the process, so that eventual reused components could be registered in the Project Management Plan.

7.1.7. Product Requirements Specification and Functional Analysis

According to the survey, the "Specify Product Requirements" and "Perform Functional Analysis" activities were considered very important in the process (see Figure 7.2). Although they were described as performed in parallel, it was suggested that these activities should be switched in the workflow since starting by building some scenarios (within the scope of functional analysis) is a common practice in the company. Likewise, the chapters in the document should be switched accordingly.

7.1.8. Peer Review

In the Peer Review activity, some confusion and concerns were revealed about its procedures. For instance, some argued that the peer review was already performed at Altran since the Project Manager reviewed the documentation produced by the functional team before handing it

to the customer. This misconception of a formal peer review meeting with the team as a simple review was explained and its importance was stressed as one of the most powerful tools to achieve quality. After this explanation all the reviewers agreed that is should be kept in the process, with an exception for service management projects. In fact, according to the survey, the respondents understood its importance as they all labeled it "very important" in the process (see Figure 7.2). Furthermore, it was suggested that a deadline date to solve the defects found should be added to the Defect Register Log.

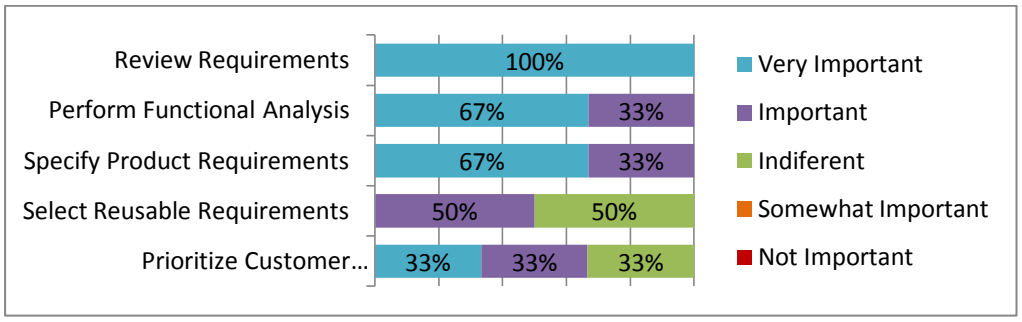


Figure 7.2 - Importance of each new activity

7.1.9. Tools and Traceability

Regarding the integration of this process with the TestLink tool, it was questioned whether the separation of Customer Requirements from Product Requirements was possible to achieve using this tool and how the Product Requirements could be grouped by their components, as in the template. In fact, as explained during the meeting, storing the information in this style is possible through TestLink's tree folder structure. However it is something that should be tested in a real project.

7.1.10. Discussion

In summary, the involvement of Altran's experts was crucial to gather feedback and raise their interest in the new process. Although everyone agreed that the procedures suggested would increase the quality of the requirements, some resistance to change was felt, especially due to their tight schedules to complete the projects. Their suggestions and concerns were taken into account and discussed internally by the authors of this thesis, causing some small changes in the first version of the proposed process, as presented in Section 6.2.

Furthermore, analyzing the survey's responses we concluded that the knowledge on the techniques proposed is average and, therefore, training in such techniques is fundamental to correctly apply them in the projects.

7.2. Case Study

The methodology created for the new process was applied in a case study to illustrate how to apply the techniques suggested in practice. This analysis of a real-life project allows evidencing the feasibility of the proposed concepts and templates and testing the most time-consuming activities of the process in a real setting.

The case study *AltranREQ* was selected from a set of Altran's past projects, as it was a small-size project, there was a previous knowledge on its application domain and some of the suggested techniques like GUI mockups, use cases and behavior diagrams, were already employed. The information provided in AltranREQ's requirements documents was further developed and adapted to the new Project Requirements Specification Template proposed in this dissertation. In this section, this case study is presented, along with some examples of the produced artifacts. The full example of the Project Requirements Specification document can be consulted in Appendix F.

7.2.1. AltranREQ Project

The overall objective of *AltranREQ* is to provide an appropriate solution for the management of requirements for IT projects. This project was performed under the scope of Altran's Academy and aids the project managers and functional teams, providing them tool support to manage and develop the requirements of Altran's Projects.

7.2.2. Customer Requirements

The identification of the concerned stakeholders' roles and the transformation of the existing high-level requirements in user stories were straightforward. Table 7.1 exemplifies one of the user stories created.

Table 7.1 - Example of a Customer Requirement in the format of a User Story

CR 04-F	*User Story Export Information*
Description:	*As a **Project Manager**,* *I would like to export all elements related to a project,* *So that I can print them and present them directly to the client.*
Priority:	*SHOULD*

7.2.3. Functional Analysis

Some of the artifacts suggested to analyze the work product functionality were already provided in the requirements documentation previously produced. In particular, for the **Functional Description** section, the use cases and the sequence diagrams proposed for the scenario specification were adopted. However, the use cases had to be slightly modified due to the incorrect use of some of the templates' fields. The sequence diagrams were adopted to identify all the sub-functions in each scenario. As described in the template, activity diagrams could also be used for this purpose.

The existing documentation did not provide a technique for quality attributes. The innovative technique proposed to analyze quality attributes, the Quality Attribute Scenarios, were used to describe such attributes using the quality attribute catalog proposed and based on the authors own expertise. Figure 7.3 exemplifies one of the created Scenarios.

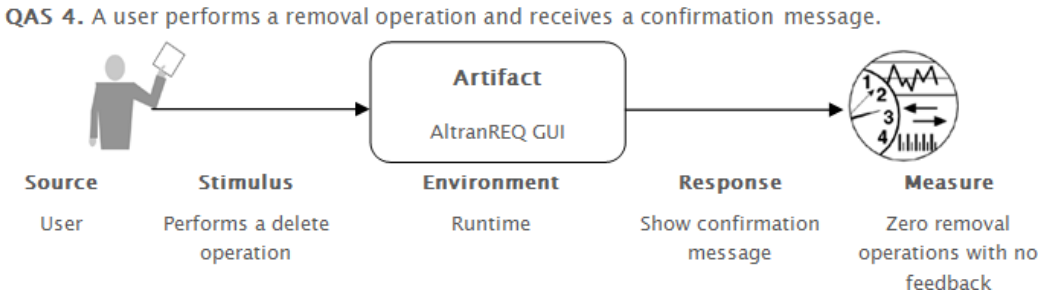


Figure 7.3 - Example of Quality Attributes Diagram

Likewise, the **Functional Architecture** section was also not provided by the previously existing documentation. Therefore, all its artifacts were created from scratch and decisions were made according to the authors' judgment. Regarding the Quality Attribute Trade-off Analysis, we decided, for this particular project, that:

- Measures to prevent authorized access, like password and decipher a visually encrypted code, improve Security but hurt Usability as they make the system less simple to use.
- Exchanging information with another system through a web-service is good for interoperability, but might present security threats on the exchanged information. Plus, using standard communication protocols will improve future maintainability.

Tables 7.2 and 7.3 were used to analyze the Quality Attribute trade-offs and their relationship with the logical components.

Table 7.2 - Quality Attributes Contribution Table

	Security	Usability	Interoperability	Maintainability
Security		-		
Usability				
Interoperability	-			+
Maintainability				

Table 7.3 - Quality Attribute's impact on logical components

	Administration	Exporting	History	SearchEngine	Users	ProjectInfo
Security	X				X	X
Usability				X		X
Interoperability					X	
Maintainability	X					X

The logical architecture of the system was also not provided in the previous documentation and was created using a UML component diagram (see Figure 7.4), which illustrates the logical components, their internal interfaces and the dependencies between them.

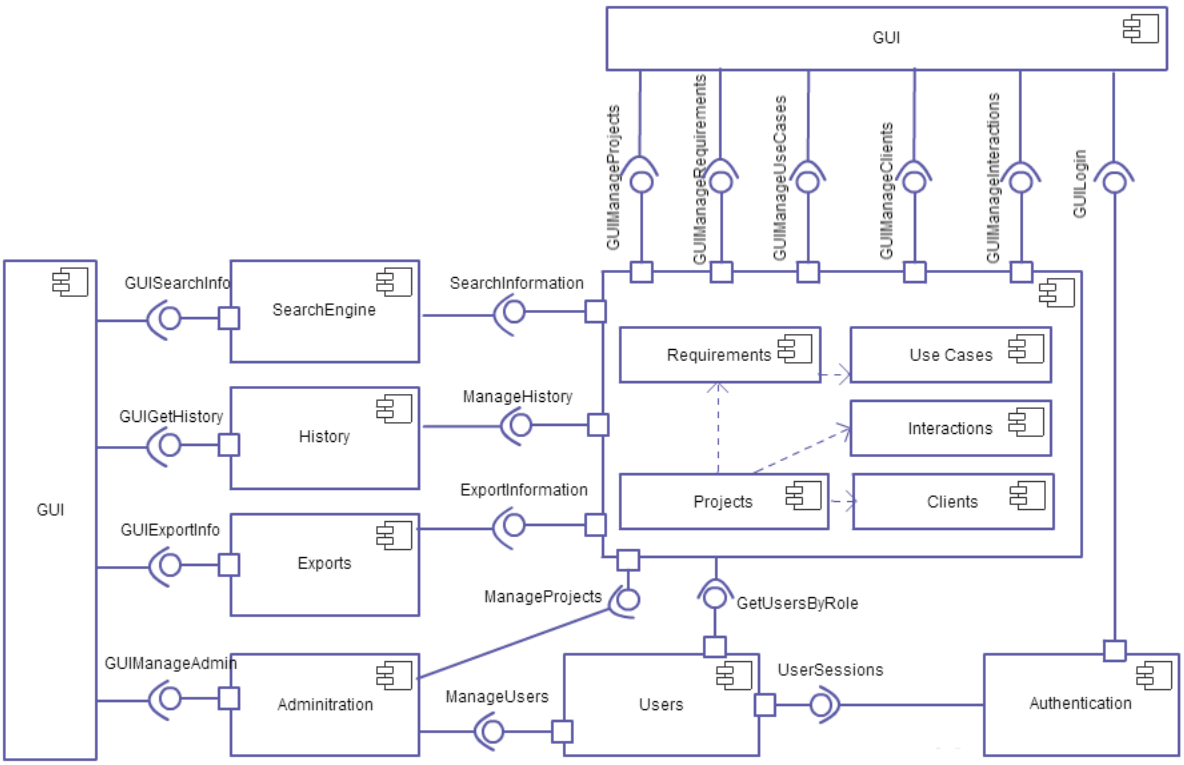


Figure 7.4 - AltranREQ Logical Component Diagram

The abstraction level chosen for the components is subjective and, therefore, was based on the authors' judgments. To evidence the possibility of different levels of detail, we illustrate the encapsulation of the internal components "Projects", "Requirements", "Use Cases", "Clients" and "Interactions" in a single component.

The Graphical **User Interface** Mockups were specified in the previously produced requirements documentation. We reused the mockups provided and extended each interface presented with a table of the interactive features (for example buttons and tabs) of each screen, as shown in Table 7.4.

Table 7.4 - Example table describing interactive features of SREEN 1 "Project Details"

GUI Object	*Type*	*Action*	*Notes*
Exportar	Button	Go to SCREEN 2 – Export Confirmation	Downloads project information in word document.
Guardar	Button	Go to SCREEN X – <Name>	Save edited project information.
Cancelar	Button	Go to SCREEN Y – <Name>	Cancel changes made on the project info.
Administração	Tab	Go to SCREEN Z – <Name>	Administration operations.

7.2.4. Solution Requirements

The Solution Requirements were built in parallel with the functional analysis. The identified components were detailed with their specific Solution Requirements. To illustrate this component specification, the component "Exporting" was selected and its functional, quality attribute, interface and technical requirements described. These low-level requirements were not provided by the previous documentation. Table 7.5 exemplifies one functional Solution Requirements that belongs the "Exporting" Component.

Table 7.5 - Example of a Solution Requirement of Component "Exporting" (COMP 1)

SR 1.01 - F	*Export Project Information*
Use Case:	UC 01Export All Information, UC 02 Export Functional Requirements, UC 03 Export Use Cases, UC 04 Export Non-Functional Requirements
Mockup :	SCREEN 1 – Project Details
Description:	The application should provide a mechanism to export the selected project information into a printable document: • Use Cases • Functional Requirements • Non-Functional Requirements • Integral Project information
Source:	Altran
Evaluation Criteria:	Obtain a document containing the information selected to export.
History:	Original version 1.0 18/08/2014

Those Solution Requirements that are common to several components, i.e. transversal to the application, were placed in the **Crosscutting Solution Requirements** section. As displayed in this section, quality attributes and technical requirements are common types of requirements that can affect several components. Most of the quality attribute requirements were adopted by the existing documentation.

Regarding the **Data Requirements**, the domain model and the specification of the domain entities were reused from the previous requirements specification.

7.2.5. Traceability

To illustrate how the traceability between the Customer and Solution Requirements can be achieved, a traceability matrix was created. Table 7.6 provides a partial view of such traceability matrix. The remaining artifacts can also be traced, as they are associated to the respective Solution Requirements in the document.

Table 7.6 - Partial view of the traceability matrix created

Customer Requirements	Components	Solution Requirements
CR 04 - F	COMP 01	SR 1.01 – F
		SR 1.02 – F
		SR 1.03 – QA
		SR 1.04 – I
		SR 1.05 – T
CR 02 - NF	System	SR 01- QA

7.2.6. Discussion

In summary, the application of the new methodology proposed in the selected case study shows that the suggested methods and techniques can indeed be employed in a real project. The case study selected was one of Atran's past projects and, therefore, some process activities could not be conducted. Given that this implementation in a practical case intended to be merely illustrative, it was only partially applied. A specific Customer Requirement was selected and developed accordingly, throughout the requirements template. Some sections of the template ("Operational Concepts", "Prerequisites" and "Terminology") were not illustrated because they were considered not applicable to the selected case study, what is considered a limitation of the chosen example.

Besides the methodology and techniques applicability, no further validation conclusions can be drawn to avoid biased results. Since most of the artifacts were created by the authors themselves, drawing any conclusion could be considered as a major threat to internal validity. As described in the next Section, an application independent from these authors was also conducted.

7.3. Pilot Project

The process developed should be piloted in a "live-test" project with an independent team of consultants to ensure their acceptance and organizational suitability. This pilot phase is critical because it will allow drawn the most important conclusions regarding the process validation. Next, the criteria for selecting our pilot project, its implementation and results are described.

7.3.1. Identify Pilot Projects

The set of candidate projects for piloting must conform to specific criteria: the selected projects must be closed, small sized projects, in the beginning of their life cycle, so that the new requirements process can be tested throughout and within the time frame of this dissertation validation period; the team should have at least one staff member with some experience on the current processes; the requirements analysis phase should be short enough so that some conclusions can be presented in the time given for the writing of this dissertation.

7.3.2. Establish Success Criteria

To determine if the project being piloted with the new process is successful, the evaluation criteria should be defined. These criteria will be based on the duration of the project, the applicability of the process' procedures and the feedback gathered from the consultants. The pilot project will be considered successful if its duration is within the estimated time and the procedures are applied with little effort.

7.3.3. Implement the Pilot

The requirements for candidate pilot projects were presented to Altran's management and project P[4] was selected to test the new process. Project P was a closed, small sized IT project in the domain of aviation manufacture. This project was developed at Atran's Nearshore Center at Fundão and its requirements phase was the responsibility of one Project Manager and one experienced functional consultant. As they were geographically distant, the team applied the methodology proposed, with its process and techniques, independently, hence avoiding bias.

Although the requirements phase had already officially started when the project was proposed for piloting, it was decided that it would still be the best project option, given its small size and expected duration within our temporal limitations. Therefore, project P was piloted to test specific activities of the process. In particular, the following core activities were tested:

- Define and Prioritize Customer Requirements
- Perform Functional Analysis
- Specify Solution Requirements and Test Cases
- Review Requirements (Peer Review)

The new Requirements Management and Development process was briefly described to the functional consultant over a phone conversation and the new templates created were made available.

7.3.4. Analyze Results

As initially estimated, the requirements phase of the pilot project lasted two weeks with the applicability of the new procedures tested in the proposed process activities by the functional consultant.

7.3.4.1. Define and Prioritize Customer Requirements

The Customer Requirements were properly registered in the template in the format of user stories using the MoSCoW method for prioritization, as suggested in the process. Four stakeholders were identified and briefly described, followed by the definition of 24 user stories, 18 of them functional and 6 non-functional. While completing this first chapter of the template, we noticed that there was a minor misconception of the stakeholder's roles as the roles of the team developing the project. This was pointed out to the consultant and quickly corrected. No further issues were detected.

7.3.4.2. Perform Functional Analysis

Regarding the Functional Analysis, most of the artifacts were generated successfully. Functionality was described through 51 use cases using the template provided correctly. Likewise, 6 quality attribute scenarios (one for each quality attribute in the Customer Requirements) were specified as in the example. The main screen of the Graphical User Interface was briefly sketched and the remaining screens of the application were indicated as specified in a separate document. However, no details about the interactions with the objects in the screen were provided. Furthermore, the Functional Architecture with a component diagram and the analysis of scenarios using behavior diagrams were disregarded.

[4] The name of the Project will not be revealed here for confidential purposes

7.3.4.3. Specify Solution Requirements and Test Cases

The Solution Requirements were a concept that was not fully understood by the team as they were not provided in the project requirements template. When asked about the absence of these detailed requirements, the team claimed that these were specified in a separate document as "technical requirements", which should not be mixed with the functional specification. Nevertheless, the data requirements section, with the domain model and the specification of the domain entities was indeed provided.

7.3.4.4. Review Requirements (Peer Review)

The pilot project team decided to perform this activity informally in a meeting attended by the project manager, the functional consultant and a client representative. No further members of the project team were able to attend the review meeting due to schedule restrictions. The requirements documentation was in fact reviewed. However, the log of the defects found was not recorded.

7.3.5. Discussion

Analyzing the results of these activities, we can conclude that two major problems were found: (i) lack of specification of the functional components and (ii) misinterpretation of the Solution Requirements as technical requirements of the architecture. The team justified the absence of these artifacts in the project requirements specification with the same reason. They claim that both should be part of a technical document because the requirements specification doesn't include technical details. Although this is in fact a true affirmation, neither the logical components nor the Solution Requirements are intended as technical details of the architecture.

The identification of logical components attempts to group requirements by their similar functionality and illustrate how these functional modules interact with each other, so that the dependencies between them can be exposed. This Functional Architecture is proposed in the functional analysis in a component-based development perspective, and should not be mixed with the technical architecture presented in the technical document. Likewise, the Solution Requirements were misinterpreted. They are intended as the low level requirements of the identified logical components, which can be functional, quality attributes and interface or technical restrictions proposed by the customer.

Besides, the separation between Customer and the Solution Requirements represents an important goal of the CMMI Requirements Development process area. The technical document created is indeed necessary for the project design, but it is related to another process area of the CMMI (Technical Solution), which is closely related to the requirements development, but is out of the scope of this dissertation.

The problems found reveal that there is a general need of training in CMMI concepts and in the process' procedures proposed before the pilot projects and, specially, before implementing the process in the company. As demonstrated by this pilot, such training cannot consist of simply handing the written documentation to the consultants or over brief phone conversations. An ongoing effort with a specialized quality team is necessary to ensure the concepts are understood and applied properly. Furthermore, although the term "Solution Requirement" was proposed by Altran's Project Managers (see Section 7.1), its misinterpretation in the pilot project could indicate that this term is ambiguous and influenced the consultant to reason about the technical solution. Therefore, other terms should be considered by Altran, for instances "Software Requirement" or "System Requirement".

7.4. Lessons Learned and Threats to Validity

The overall findings and conclusions of this validation phase may be threatened by some factors. First, the reduced number of participants in the feedback meeting makes the sample of Altran's experts not representative. Regarding the case study, it was applied by the authors of the process as an illustration of some core process' activities. Therefore, it can be considered

bias. Also, the pilot project was not directly monitored by the authors as initially planned, which might have difficult gathering data for its analysis, but prevented biased results.

The main lessons learned while validating the new Requirement Management and Development process through these three approaches are:

- **The practices proposed in the process are applicable to Altran's projects**. Although the positive impact in requirements quality was acknowledged, Project Managers expressed their concerns on adding more requirements practices due to their tight schedules.
- **Differentiating Customer from Product Requirements** as proposed in CMMI is a major challenge because most projects are driven exclusively by the Customer Requirements.
- **Training project teams in CMMI concepts** in general and in the processes procedures in particular, is a clear need before implementing the process.
- **A Quality Team responsible for CMMI implementation and maintenance** is essential to provide this training and coordinate improvement activities. This team should have the right skills, experience and motivation for leading process improvement.
- **Senior Management Commitment** is critical, especially in the pilot and implementation phases.
- **Processes must be practiced and continuously improved** to accomplish the organization's business goals and achieve the CMMI advantages.

7.5. Summary

Throughout this chapter, the new requirements management and development process proposed in this dissertation was validated using three different approaches. First the opinions of Altran's experts on the first version of the process were collected and the process was refined accordingly. Then, the application of the requirements template was illustrated using a selected case study. Finally, and most importantly, a pilot project was used as a live-test on the process core activities.

Analyzing the results of this validation we can conclude that the process is viable and the methods proposed are indeed applicable to Altran's context. However, as demonstrated by the pilot project, there is a clear need for training project teams in CMMI concepts, its importance and in the procedures proposed in the process. Handing the written documentation to the teams is not enough. A quality team responsible for conducting this training and to supervise the piloting and implementation phases is highly recommended. We believe that this kind of commitment is important to take advantage of the quality improvements of the new process and these, on the other hand, are fundamental for the CMMI-level 3 certification.

8. Related Work

The implementation of CMMI Requirements Management and Development best practices in organizations is not a topic often addressed by the research community. To review the state of the art on this subject, we adopted a method inspired on Kitchenham guidelines for systematic literature reviews [92].

The search terms and respective synonyms used to identify relevant articles were '*CMMI'*, '*Capability Maturity Model Integration'*, *'CMMI Requirements Engineering'*, *'CMMI Requirements Development*', '*CMMI RE'*, '*CMMI RD'*. Such terms were searched in six digital libraries (IEEExplore, ACM, ScienceDirect, Google Scholar, DBLP, Springlink) selected for their relevance in Software Development and Information Systems. In addition, the proceedings of the III CMMI Portugal conference were consulted and the references of the relevant articles were also checked.

Due to the time constraints and the informality of the review, it was necessary to ensure that only the most relevant articles were analyzed. Therefore, an inclusion criterion based on the content of the publication was created: Studies that relate CMMI and RE; Studies that perform CMMI-based assessments of RE processes; RE methodologies compliant with CMMI. All articles about other process areas, other maturity models and high maturity levels were excluded. We also excluded studies about the compliance of CMMI with Agile technologies and Software Product Lines as they are not relevant for the context of Altran Portugal.

Nearly 120 articles were found. Several authors have analyzed the topic of RE maturity, creating new RE maturity models based on CMM(I) (e.g., [8], [93], [94]). Others focused on other process areas (e.g., [95], [96]) or integration of CMMI with other maturity models (e.g., [97], [98]). Studies related to the implementation of CMMI high maturity levels (e.g. [99], [100]) or related to the compliance of Agile methodologies with CMMI (e.g. [101], [102]) can also be found in the literature. However, results from these studies are not related to the scope of this dissertation. After the excluded articles, six relevant studies were selected, analyzed and classified according to two research topics: CMMI-based Assessments, presented in Section 8.1, and CMMI-based Methodologies, in Section 8.2. Section 8.3 provides a summary of this Chapter.

8.1. CMMI-based Assessments

Vasconcelos *et al.* [103] claim that most works related to CMMI compliance assessments of software development processes encompass some weaknesses, since the analysis are not detailed and not based on the SCAMPI method. In fact, the works of [104], [105] and [106] study the compliance of RE processes with the RD practices of CMMI, but fail to explain how an approach complies with the model, where it conflicts or where adjustments should be made.

Two relevant CMMI-based assessments relative to the RD process area were found and will be described and analyzed next.

8.1.1. Requirements Development at ALERT

In his MSc dissertation [51], Espinheira improved the requirements development methodology used by ALERT Life Sciences Computing company through the application of CMMI RD best practices as guiding standard. This company is focused on the development of healthcare software for clinical environments. Although the application domain is very different from Altran Portugal, the methodology used to assess the current situation of the company and to develop a new process compliant with RD process area is quite relevant for this dissertation purposes and therefore should be analyzed.

To improve the company processes, Espinheira started by identifying and detailing the current workflows and templates used at ALERT. These processes were then analyzed using customized SCAMPI C based evaluation to detect a set of non-compliances and out of the scope issues with the CMMI RD best practices.

This compliance analysis was performed through a bottom-up approach, using the low level CMMI components to rate the RD specific practices. The subpractices and typical work products of each RD specific practices were mapped against the company's workflows and templates. The specific goals were not rated due to the informality of the assessment.

The subpractices and typical work products were rated using a simple binary scale (satisfied, unsatisfied) considering the possibility of some *"out of the scope"* situations. The specific practices were rated using the following scale based on SCAMPI A:

- Not Implemented (0 to 15% of subpractices and work products satisfied)
- Partially Implemented, (>15% to 50% of subpractices and work products satisfied)
- Largely Implemented (>50% to 85% of subpractices and work products satisfied)
- Fully Implemented (>85% to 100% of subpractices and work products satisfied)

The current methodology of ALERT was evaluated using a matrix, showing the rate assigned to each typical work product and each subpractice of RD. Although the rationale behind the subpractices' rating is not provided, a brief description of how each specific practice is addressed by the company is given. The outcomes of this analysis were gathered on a SCAMPI result table (Figure 5.1) to allow a general overview of the evaluation and highlight the areas where improvements can be made.

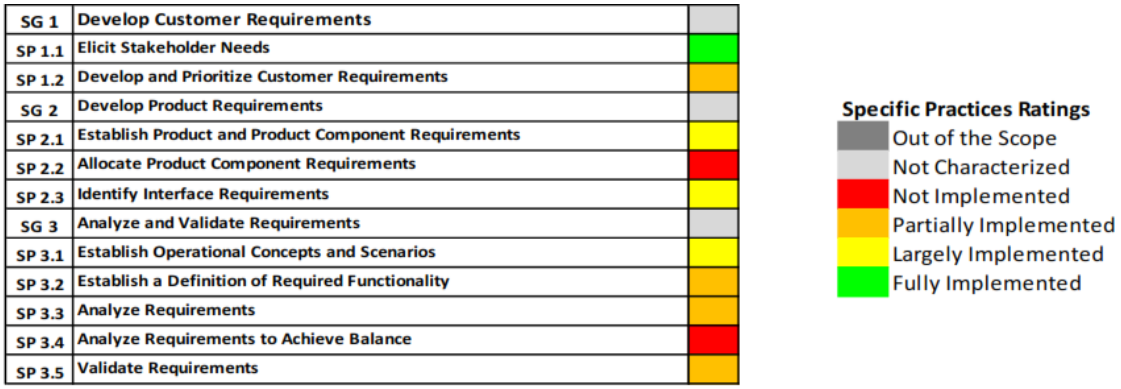


Figure 8.1 - Requirements Development Compliance Evaluation, taken from [51]

To reduce the non-compliances found in the assessment, Espinheira proposed the following improvements: creation of two additional teams to aid in the activities of prioritization and validation; separation of the current workflows in stages to reflect the distinction between customer and product requirements; update of the current templates with new chapters, such as Stakeholders, Customer's needs, Component Specification; Prioritization of needs with the MoSCoW rules (Must, Should, Could, Would).

The proposed methodology was then evaluated using the same SCAMPI based evaluation explained previously. To understand and measure the impact obtained, the "before" and "after" results were combined in a SCAMPI result table. Additionally, the new methodology was also applied and described in a real project. The results show that, although only two practices were evaluated as "Fully Implemented", the proposed changes had a positive overall impact among teams and customers. Even if there was still room left for improvements, Espinheira's work successfully brought the company closer to CMMI best practices.

8.1.2. Requirements Development with BPRE4OO

Vasconcelos *et al.* [103] also conducted a SCAMPI based compliance assessment to the RD process area. The aim of their study was to evaluate how the business process-based RE

approaches comply with CMMI, so that they can be used by industry. The RD process area was chosen since it is the one that targets requirements elicitation and specification, as business process-driven RE approaches do.

To perform this analysis, they chose a specific business process-driven RE approach developed by some of the authors – the BPRE4OO (Business Process-driven RE for Object-Oriented conceptual modeling) – and compared its features against the RD best practices.

Similarly [51], the compliance analysis was performed bottom-up. However, in this study the typical work products and subpractices were not directly evaluated. The assessment was done from practices to goals, using two types of objective evidence, adapted from SCAMPI guidelines: Affirmations and Artifacts. Even though a typical SCAMPI evaluation is usually performed using artifacts from actual projects, Vasconcelos *et al.* defined an assessment method based on the BPRE4OO documentation because the approach was never applied in industry.

The characterization scale used in the study to rate the implementation of practices was based on SCAMPI A, and is summarized in Table 5.1.

Table 8.1 - practices characterization scale, based on [103]

Fully implemented (FI)	Evidences are present and judged to be adequate for demonstrating the implementation of a practice, and no weaknesses are found.
Largely implemented (LI)	Evidences are present and judged to be adequate for demonstrating the implementation of a practice, but some weakness is found.
Partially implemented (PI)	Although some information suggests that aspects of the practice are implemented, some or all the data required are absent or judged to be inadequate, and some weakness is found; or the data supplied to the assessment team present conflicts (i.e., certain data indicate that a practice is implemented and other that it is not) and some weakness is found.
Not implemented (NI)	Some or all the data required are absent or judged to be inadequate, the data supplied do not support the conclusion that the practice is implemented, and some weakness is found.

Based on the grades of their associated practices, the rating of each specific goal was done using a binary scale, similar to the scale used in SCAMPI A appraisals:

- ***Satisfied***: If and only if all the associated practices are graded as either largely implemented or fully implemented, and the aggregation of the weaknesses of the practices does not have a significant negative impact on goal achievement.
- ***Unsatisfied***: If at least one of the associated practices has a grade different from largely or fully implemented.

The compliance analysis details are presented in the study, providing a description, artifacts, affirmations and a grade to each specific practice. With the practices graded, each specific goal was also rated as “Satisfied” or “Unsatisfied”. To conclude, the capability level of the process was assessed. Because at least one of the goals was “Unsatisfied”, the RE process of BPRE4OO was evaluated with capability level 0.

Based on the weaknesses found, the authors gathered a set of improvement suggestions organized by practice and stage of BPRE4OO, as exemplified by Table 5.3. Most of the improvements suggested were simple adjustments made to the process, related to explicit modeling and documentation of evidences.

Table 8.2 - Improvement suggestions, based on [103]

Improvement	Stage	SP
Register requirements changes and inclusion of mechanisms for traceability between customer and product requirements.	Spec. of system reqs.	SP 2.1

According to the authors, the results of the assessment for this approach can also be generalized to other business process RE approaches, since most of them share the same characteristics.

8.1.3. Analysis of the Approaches

The approaches analyzed in this section are relevant for the scope of this dissertation because they both provide SCAMPI based assessments relative to the RD process area. Through this study it was possible to understand how the SCAMPI guidelines [36], [37] are usually applied to perform assessments of processes, specifically for the RD process area. Additionally, it was possible to realize that most improvements made to the current requirements processes are based on small changes. However, these changes have a major impact in the rating of specific practices and, consequently on the benefits achieved by the organization.

The analysis of the assessment methods used is summarized in 5.3.

Table 8.3 - Analysis of the assessment approaches

	ALERT	BPRE4OO
Domain	Healthcare software development company	Business process-driven RE approach
SCAMPI type	C	A
Rate of Goals		x
Rate of Practices	x	x
Rate of subpractices and work products	x	
Improvements	Creation of two additional teams. Separation of the current workflows in different stages. Addition of sections to the templates.	Modeling of req. for all phases. Document changes and traceability. Register evidences for V&V. Explicit conflict resolution.
Strengths	Use of SCAMPI results table. New processes applied on projects.	Explicit use of object evidence to support the rates given.
Weaknesses	Lack of rationale to justify subpractices grades.	Improvements made were not tested on actual projects.

The major difference between the two studies is the type of SCAMPI used. On the one hand, the work of Espinheira [51] aimed at improving the current processes of a company using CMMI guidelines through an informal assessment. Subpractices were rated, deriving the grade given to their associated practices. Espinheira chose this assessment because of the scope, time and his experience with both the process being evaluated and the CMMI model. On the other hand, Vasconcelos *et al.* [103] preferred an appraisal based on SCAMPI A to determine the satisfaction of the goals and, therefore, the capability level of the process. The authors had a lot of knowledge about the process being evaluated and had also previous experience with CMMI.

8.2. CMMI-based Methodologies

Research on CMMI-based methodologies compliant with the RD process area is also partially relevant to this dissertation, given that some ideas might be adopted when proposing an improved process for Altran Portugal.

The proposed RE process in the work of Cerón *et al.* [104] is not applicable to the context of this dissertation. Although compliant with the RD process area, this process was design to attend the specific needs of system families engineering, where a great degree of reuse of requirements exists for the development of similar systems. As Altran is a consulting company, their projects are very diverse and in multiple application domains. Therefore, processes centered in reuse of requirements don't fit the particular needs of Altran's projects.

In [107], Wang *et al.* propose a model compliant with the best practices of the RD process area. The authors claim that this model develops the requirements from goal, use-case and scenario viewpoints, based on the "*recommended process"* of the RD process area. However, looking at

the CMMI specific practices as a process or process' description rather than a process characteristic is one of the most common mistakes reported by the literature [20, 48].

The proposed model identifies stakeholders and elicits their needs using a set of questionnaires and templates. The responses to these questionnaires are then analyzed with a quality policy and consolidated using a statistical method for information classification.

The Customer Requirements are managed using two sub-components of the model. The Customer Requirement Derivation Model (CRDvM) helps to ensure traceability of requirements, the use of change proposals to guide modifications and to development requirements according to the business goals. The Customer Requirement Decision Model (CRDcM) supports the project team in resolving conflicts by evaluating alternatives and recording the rationale for the decisions. To develop the product requirements, the model proposed the division of requirements into components identified with the support of a questionnaire. As every business domain has its own language, the study also proposed the elaboration of a Glossary of specific business terms.

The study of Yoshidome *et al.* [109] proposes a methodology based on a set of free software tools that support the implementation of a Requirements Development process compliant with CMMI. The following tools were suggested:

- **OSRMT –** Requirements Management tool used to record and describe requirements
- **Openproj** – Project Management tool customized by the authors
- **Redmine –** Online tool used for bug tracking and control the progress of tasks
- **Spider-CL –** Tool used to create checklists
- **Astah Community–** UML modeling tool

The aim of this study was not to define a process, however the use of such tools will only remain compliant with the RD process area regarded that a proper activity flow is respected. Therefore, the authors proposed a workflow of activities based on the RD practices. They also analyzed the compliance of the tools' functionalities with each specific practice of CMMI RD process area, providing a brief explanation of how the tools are used to attain each practice.

Although this workflow and set of tools are able to address the specific practices of the RD process area, this study has some limitations. First, it was never applied in real projects. Plus, some recommended practices are not properly addressed, like performing risk analysis of requirements or driving the architectural decisions by the organizational needs and relevant quality attributes. To conclude, the authors advise the adoption of this methodology, along with the institutionalization of a defined process.

8.3. Summary

The related works found in the literature can be divided in CMMI-based Assessments and CMMI-based Methodologies. On the topic of assessments, the studies of Espinheira [51] and Vasconcelos *et al.* [103] were presented and compared regarding its strengths and weaknesses. The improvements suggestions given by each approach were not presented in detailed, as they are specific to the weaknesses found in each process and for that reason they may not be applicable to the particular needs of Altran processes. Regarding proposed CMMI-based methodologies, the RD-compliant approaches of Wang [107] and Yoshidome *et al*. [109] were briefly explained. Wang [107] proposed a model based on questionnaires, while Yoshidome *et al*. [109] adopted the use of free software tools to support a potential process.

These research topics presented can be a strong contribution to this dissertation. On one hand, the assessment approaches can be used as a basis to define the evaluation method for the processes currently used at Altran Portugal. On the other hand, existing compliant methodologies provided ideas for the improved process proposed for Altran.

9. Conclusions and Future Work

This Chapter presents a brief analysis of the work performed and goals accomplished throughout this dissertation. In Section 9.1, the path followed to solve the considered problem is summarized, along with the challenges faced and the main results achieved. Section 9.2 explores possible limitations of the work performed that were out of our reach. Potential future works related to the solution proposed are presented in Section 9.3 and finals remarks on this research experience are stated in Section 9.4.

9.1. Dissertation's Overview

Delivering high-quality software in time and at the lowest possible cost is a major challenge faced by all organizations. Typically, the central problems when balancing these factors rely in the beginning of the projects, in the requirements phase. In fact, poorly-defined requirements lead to ambiguities, which can easily escalate to later stages of the development cycle, where mistakes get harder and more expensive to fix [8]. As a consequence, systems become over budgeted, overscheduled or don't meet the real customer's needs [14]. Based on the premise that improvements in the development processes result in higher quality products, maturity models, like CMMI, emerged as SPI solutions to guide organizations in this quest for quality. Given the criticality of the requirements phase in the projects, the CMMI addresses the aforementioned issues by defining a set of best practices for Requirements Management and Requirements Development.

In this context, this MSc dissertation was a part of the efforts made by Altran Portugal to improve their current processes using the CMMI Framework. Having a maturity level 2 certification, Altran had previously implemented a Requirements Management process. In this dissertation we intended to further develop this process to comply with the guidelines of the Requirements Development process area, addressed at maturity level 3.

To solve the challenge proposed for this MSc dissertation, we started by analyzing Altran's current SGD processes and evaluate them with a SCAMPI-based assessment to find the set of non-compliances that could be improved regarding the considered process area. This evaluation exposed four main research questions that guided the investigation of alternative solutions to bridge the gaps found.

Furthermore, to evaluate the current industrial trends in Requirements Engineering and its relationship with CMMI maturity, a survey was conducted during the III CMMI Portugal conference. Although the amount of Portuguese companies appraised is rather small, the interest in CMMI maturity has been steadily rising. The survey revealed that the techniques preferred for eliciting, modeling and validating requirements are agreed by most companies. In particular, an emphasis was given on the use of simple notations, like natural language supported with UML diagrams, to represent requirements. Besides, it indicates that CMMI-certified companies seem to be more aware on the importance of requirements validation and the use of automated tools to manage requirements.

Based on the solutions found in the literature and the outcome of the industry survey, a new Requirements Management and Development process was proposed. The existing workflow was updated and extended with new activities. In particular, the Customer Requirements reflecting the stakeholders' needs were enhanced with simple prioritization techniques and the assessment of their related risks. Then, requirements reuse activities were added to the process to reduce project work and promote efficiency. A systematic approach to analyze functionality was also proposed through the definition of specific artifacts like use cases, quality attribute scenarios or GUI mockups. An activity for reviewing requirements was also considered in the format of a formal Peer Review meeting, in order to minimize the chances of defects found in

later development stages. All these activities were supported with proper templates to store the produced artifacts and aid in their creation.

The new process proposed was then subject to a new evaluation, using the same SCAMPI-based assessment defined earlier. The comparison of both appraisals' results demonstrated that the CMMI compliance level with the Requirements Development process area has indeed increased, having all of its specific practices successfully evaluated as fully or largely implemented.

Finally, in order to ensure the new process feasibility and assess its acceptance, it was validated using three different approaches: direct feedback gathered from Altran's Project Managers and Consultants, the illustration of the methodology application in a case study and the implementation of the core activities in a pilot project. The validation conducted helped to refine the developed process according to the teams' needs and confirmed its applicability in the organization. However, it also exposed some of the ongoing fragilities that should be improved. In particular, it evidenced that project teams must be trained in the procedures proposed in the process. Also, the nomination of a specialized quality team responsible for giving this training and for overseeing the process implementation is highly recommended.

Reaching the end of this dissertation, we consider that the goal was achieved: a new requirements management and developed process compliant with the CMMI Requirements Development process area was developed for Altran Portugal. The major improvements brought to the company concern the implementation of reusability, and verification and validation practices, like peer reviews and prototypes. Considering that the process will be carried out as proposed, it will, in time, lower the number of defects found in later stages, increasing the team efficiency, productivity and knowledge exchange. Besides, the definition of a systematic functional analysis with an emphasis on the influence of quality attributes in architectural decisions will improve consistency and help to predict such qualities of the final product. Despite the realization of such benefits, the increase in the number of artifacts to produce was a concern expressed by most project managers, which confirms the increase in documentation as the major disadvantage of the CMMI Framework.

9.2. Limitations

One of the main challenges faced during the writing of this dissertation was the definition of a validation method for the developed process. The use of quality metrics to evaluate the performance of the new process was considered. For instance, collecting the number of defects founds in later stages of development could be used as an indicator of quality improvements. Especially after implementing defect prevention activities like the Peer Review. The application of the new process in a previously developed project, collecting such metric was considered. However, given that quality metrics are not currently being collected for projects, a comparative analysis to demonstrate actual improvements was not possible.

The initial plan was for the authors to be involved in the pilot project, monitoring the application of the process to a real project. However, finding such project was not easy due to difficulties in synchronizing the timeline of this dissertation with that of a new project.

At the end, a candidate pilot project was made available within a month to this dissertation deadline. The project was conducted at Altran's Nearshore Center at Fundão, which prevented an ongoing direct monitoring of the project by the authors. Although this was not our initial choice, having a geographically distant team had its advantages. The fact that the project team was applying the process independently prevented them from being directly influenced by the authors and, therefore, the results are less biased. However, on the other hand, collecting feedback and data for a more formal analysis was more difficult.

9.3. Future Work

The new Requirements Management and Development process developed for Altran Portugal should be piloted across further projects with different variables, like size, business domains and project teams. As evidenced in the validation phase, these teams must receive proper training in CMMI concepts and the process procedures to raise their awareness on the subject. Also, the existence of a specialized quality team to provide the trainings and to supervise that the processes are being properly followed is of most importance. Such team should be introduced not only for the pilot and implementation phases, but in an ongoing basis to overcome a major challenge: CMMI maintenance. This would ensure that processes are actually carried out and continuously improved, so that the advantages of CMMI implementation are not lost.

9.3.1. Process Instrumentation

Finally, a tool tailored to support the developed process and designed to allow reusability should also be implemented. The AltranREQ application, used as a case study in Section 7.2, could be modified and used as a starting point to build such tool. In fact, most of AltranREQ's features could be maintained, like the registration of projects, clients, project teams, functional and non-functional requirements, use cases and the export of all this information into requirements documents.

However, this tool should be modified to support the separation of requirements in Customer and Solution levels and allow the specification of logical components, including their internal interfaces, which could be shared among projects for reusability purposes. The Customer Requirements could be registered as user stories and prioritized using an ordinal ranking or the MoSCoW method. Solution Requirements could be registered using the fields suggested in the template and grouped according to their components, similarly to the testLink folder structure. Moreover, the risks related to requirements could also be recorded and associated to the respective requirements, generating the requirements risks spreadsheet as proposed in the new process. In addition to the use cases, this tool could be extended to register Quality Attribute Scenarios. The quality attributes trade-offs would be then analyzed by the architect using this same tool, creating the contribution tables and aiding in architectural design decisions. To store all the models and GUI Mockups proposed in the process, the simpler solution would be to enhance the tool with a mechanism to import image files to the project. However, a more realistic and useful approach to reusability, should consider the creation of a repository of models, catalogs of quality attributes and patterns of design and architecture, so that the repository could be searched based on such metadata included in the models. Creating relationships between all the artifacts should also be a considered feature, so that a traceability matrix could then be exported.

This tool might also be integrated with the other tools currently used at Altran. In particular, given that TestLink allows importing data in XML format, our tool could provide the requirements to TestLink by exporting them in a compatible format. In this way, requirements could be traceable to their respective test cases, maintaining TestLink as a Test Management tool without duplicate information. Additionally, the documents generated by this tool could be directly stored in the proper folders of the Knowledge Tree.

The proposed tool would be a major advantage relative to the currently used TestLink since it would be able to properly support certain requirements analysis practices of the new process that TestLink, as a Test Management tool, does not provide. Besides, it would be particularly useful to attain the desired requirements reusability, acting as the reusable solutions' repository proposed in the new process. Using the search mechanism provided in our tool, both practice and project managers could find specific business domains or components, reusing previously validated requirements and their related models and test cases, which would greatly promote the productivity during the requirements phase.

9.3.2. Altran's CMMI level 3

One of the next goals for Altran Portugal is to achieve a maturity level 3. Therefore, their internal processes have to be further developed and evaluated regarding the generic goals and other 10 process areas on Engineering, Support, Project and Process Management. In particular, given the relationship between the Technical Solution and Product Integration process areas with the Requirements Development process area, their impact on the process proposed in this dissertation must be carefully analyzed.

9.4. Final Remarks

Developing this MSc dissertation in collaboration with Altran Portugal was a rewarding experience. It allowed me to focus in a scientific field of Software Engineering while experiencing the reality of a big organization, its development processes and major challenges. Also, the interaction with different organizational roles showed me different perspectives of people from diverse backgrounds.

As discussed in Section 9.3, our work opens the way for future dissertations in collaboration with this organization. In particular, the implementation of other process areas, or the development of a tool to automate the proposed process could be a major contribution to improve the processes' performance at Altran, helping them to achieve their business goals.

10. References

[1] I. Sommerville, *Software engineering*, 9th ed. Harlow, England: Addison-Wesley, 2010.

[2] Software Engineering Institute, "CMMI ® for Development, Version 1.3," 2010.

[3] "IEEE standard glossary of software engineering terminology," New York, NY, USA, 1990.

[4] R. Wendler, "The maturity of maturity model research: A systematic mapping study," *Inf. Softw. Technol.*, vol. 54, no. 12, pp. 1317–1339, Dec. 2012.

[5] "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," ISO/IEC 25010:2011.

[6] M. Glinz, "A Glossary of Requirements Engineering Terminology," Zurich, 2011.

[7] D. L. Gibson, D. R. Goldenson, and K. Kost, "Performance Results of CMMI ® -Based Process Improvement," 2006.

[8] G. Kotonya and I. Sommerville, *Requirements engineering*. Chichester, UK [etc.]: John Wiley & Sons, 1998.

[9] I. F. Alexander and R. Stevens, *Writing better requirements*. London [etc.]: Addison-Wesley, 2002.

[10] P. Zave, "Classification of research efforts in requirements engineering," *ACM Comput. Surv.*, vol. 29, no. 4, pp. 315–321, 1997.

[11] R. S. Pressman, *Software engineering : a practitioner's approach*, 7th ed. The McGraw-Hill Higher Education, 2009.

[12] B. H. C. Cheng and J. M. Atlee, "Research Directions in Requirements Engineering," in *2007 Future of Software Engineering*, 2007, pp. 285–303.

[13] R. Young, *The requirements engineering handbook*. Artech House, 2004.

[14] B. Nuseibeh and S. Easterbrook, "Requirements Engineering : A Roadmap," in *Conference on The Future of Software Engineering*, 2000, p. 10.

[15] Standish Group, "CHAOS Report," 1995.

[16] A. Davis, *Just enough requirements management: Where Software Development Meets Marketing*. Dorset House, 2005, p. 240.

[17] L. Chung, J. D. P. Leite, J. Cesar, and P. Leite, "On Non-Functional Requirements in Software Engineering," *Concept. Model. Found. Appl.*, pp. 363–379, 2009.

[18] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, 1st ed. West Sussex, England: John Wiley & Sons, 2009.

[19] L. L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, Non-functional requirements in software engineering, 1st editio. Boston: Kluwer Academic Publishers, 1999.Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*, 1st editio. Boston: Kluwer Academic Publishers, 1999.

[20] S. O'Brien, "Controlling Controlled English. An analysis of several controlled language rule sets," in *Proceedings of EAMT-CLAW*, 2003.

[21] J. R. Abrial, S. . A. Schuman, and B. Meyer, "Specification language," *Constr. Programs, Cambridge Univ. Press*, 1980.

[22] K. Lano, *The B Language and Method: A guide to Practical Formal Development*. Springer-Verlag London Ltd., 1996.

[23] Object Management Group, "Unified Modeling Language™ (UML®)." [Online]. Available: http://www.uml.org/. [Accessed: 03-Feb-2014].

[24] E. S. Yu, "Social Modeling and i * 1 Why Social Modeling," vol. 5600, no. c, 2009.

[25] A. Rashid, A. Moreira, and J. Araújo, "Modularisation and Composition of Aspectual Requirements," in *Proceedings of the 2Nd International Conference on Aspect-oriented Software Development*, 2003, pp. 11–20.

[26] E. Baniassad and S. Clarke, "Theme: an approach for aspect-oriented analysis and design," *Proceedings. 26th Int. Conf. Softw. Eng.*, 2004.

[27] J. Whittle, P. Jayaraman, A. Elkhodary, A. Moreira, and J. Araújo, "MATA: A unified approach for composing UML aspect models based on graph transformation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5560 LNCS, pp. 191–237.

[28] G. Mussbacher, D. Amyot, J. Araújo, and A. Moreira, "Requirements modeling with the aspect-oriented user requirements notation (AoURN): A case study," in *Lecture Notes in Computer*

Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2010, vol. 6210 LNCS, pp. 23–68.
[29] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of Computer Programming*, vol. 20, no. 1–2. pp. 3–50, 1993.
[30] A. Van Lamsweerde and E. Letier, "From object orientation to goal orientation: A paradigm shift for requirements engineering," *Radic. Innov. Softw. Syst. Eng.*, no. I, p. 15, 2004.
[31] M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, "Capability Maturity Model for Software, Version 1.1," Pittsburgh, Pennsylvania 15213, 1993.
[32] Software Engineering Institute, "CMMI® for Acquisition, Version 1.3 CMMI-ACQ, V1.3," 2010.
[33] Software Engineering Institute, "CMMI® for Services, Version 1.3 CMMI-SVC, V1.3," 2010.
[34] CMMI Institute, "CMMI Levels," 2013. [Online]. Available: http://cmmiinstitute.com/cmmi-solutions/cmmi-appraisals/cmmi-levels/. [Accessed: 17-Dec-2013].
[35] S. Peldzius and S. Ragaisis, "Investigation Correspondence between CMMI-DEV and ISO/IEC 15504," *Int. J. Educ. Inf. Technol.*, vol. 5, no. 4, 2011.
[36] SCAMPI Upgrade Team, "Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.3: Method Definition Document," 2011.
[37] W. Hayes, G. Miluk, L. Ming, M. Glover, and Members of the SCAMPI B and C Project, "Handbook for Conducting Standard CMMI Appraisal Method for Process Improvement (SCAMPI) B and C Appraisals, Version 1.1," Pittsburgh, Pennsylvania, 2005.
[38] K. Keller and B. Mack, "Maturity Profile Reports," *CMMI Institute*, 2013. [Online]. Available: http://cmmiinstitute.com/assets/presentations/2013SepCMMI.pdf.
[39] M. Staples, M. Niazi, R. Jeffery, A. Abrahams, P. Byatt, and R. Murphy, "An exploratory study of why organizations do not adopt CMMI," Jun. 2007.
[40] I. Margarido, "CMMI challenges of V1.3 and of having a new home," in *CMMI Portugal Conference Series*, 2013.
[41] I. Margarido, "Depois da III Conferência CMMI Portugal e SEPG Europe, notícias do CMMI Institute," *CMMI Portugal*, 2013. [Online]. Available: http://cmmiportugal.blogspot.pt/2013/11/depois-da-iii-conferencia-cmmi-portugal.html.
[42] University of Duisburg-Essen, "Requirements Engineering: Foundation for Software Quality (REFSQ)," *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2013. [Online]. Available: http://refsq.org/. [Accessed: 26-Jan-2014].
[43] L. Buglione and J. Hauck, "Hybriding CMMI and Requirement Engineering Maturity & Capability Models-Applying the LEGO Approach for Improving Estimates.," *ICSOFT*, 2012.
[44] S. Nivas, "Inter Relationships between Requirements Management and Requirements," vol. 1, pp. 1–4, 2005.
[45] D. Linscomb, "Requirements Engineering Maturity in the CMMI," *J. Def. Softw. Eng.*, pp. 25–28, 2003.
[46] Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. Project Management Institute, 2000, p. 216.
[47] Altran Portugal S. A., "Requirements Management Process." pp. 1–20, 2012.
[48] Altran Portugal S. A., "Execution Process." pp. 1–19, 2013.
[49] Altran Portugal S. A., "Project Management Process." 2013.
[50] WebFinance Inc, "Business Dictionary." [Online]. Available: http://www.businessdictionary.com/definition/gap-analysis.html. [Accessed: 03-Jan-2014].
[51] E. Espinheira, "A Requirements Development Methodology Compatible with CMMI," Faculdade de Engenharia da Universidade do Porto, 2009.
[52] I. Sommerville, *Software engineering*, 5th ed. Wokingham, England: Addison- Wesley, 1996.
[53] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin, "A systematic literature review of software requirements prioritization research," *Inf. Softw. Technol.*, Feb. 2014.
[54] Q. Ma, "The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements : A Systematic Literature Review," Auckland University of Technology, 2009.
[55] R. B. Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, and A. Aurum, "Prioritization of quality requirements: State of practice in eleven companies," *2011 IEEE 19th Int. Requir. Eng. Conf.*, pp. 69–78, Aug. 2011.
[56] P. Berander and A. Andrews, "Requirements prioritization," *Eng. Manag. Softw. Requir.*, 2005.
[57] T. L. Saaty, "The Analytic Hierarchy Process," *Education*, pp. 1–11, 1980.
[58] K. A. Khan, "A Systematic Review of Software Requirements Prioritization," no. October, 2006.
[59] J. Karlsson, C. Wohlin, and B. Regnell, "An evaluation of methods for prioritizing software requirements," *Inf. Softw. Technol.*, vol. 39, no. 14, pp. 939–947, 1998.

[60] S. Hatton, “Early Prioritisation of Goals,” in *Conference on Advances in Conceptual Modeling: Foundations and Applications*, 2007, pp. 235–244.
[61] V. Ahl, “An experimental comparison of five prioritization methods,” Blekinge Institute of Technology, 2005.
[62] P. Berander and M. Svahnberg, “Evaluating two ways of calculating priorities in requirements hierarchies – An experiment on hierarchical cumulative voting,” *J. Syst. Softw.*, vol. 82, no. 5, pp. 836–850, May 2008.
[63] M. Vestola, “A Comparison of Nine Basic Techniques for Requirements Prioritization,” *Helsinki Univ. Technol.*, 2019.
[64] L. Chung, B. Nixon, and E. Yu, “Using non-functional requirements to systematically select among alternatives in architectural design,” *1st Int. Work. Archit. Softw. Syst. - Coop. with 17th Int. Conf. Softw. Eng. ICSE 1995*, 1995.
[65] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Addison-Wesley, 2003.
[66] D. Ameller, C. Ayala, J. Cabot, and X. Franch, “How do software architects consider non-functional requirements: An exploratory study,” *2012 20th IEEE Int. Requir. Eng. Conf.*, pp. 41–50, Sep. 2012.
[67] J. de la Vara and K. Wnuk, “An Empirical Study on the Importance of Quality Requirements in Industry.,” *SEKE*, vol. 3520, pp. 3–8, 2011.
[68] A. Moreira, “NON- FUNCTIONAL REQUIREMENTS (& QUALITY ATTRIBUTES).” Lecture presented in Arquiteturas de Software. Universidade Nova de Lisboa, Portugal, 2012.
[69] D. Ameller, C. Ayala, J. Cabot, and X. Franch, “Non-functional Requirements in Architectural Decision Making,” *IEEE Softw.*, vol. 30, no. 2, pp. 61–67, 2013.
[70] Atlantic Systems Guild Ltd, “Volere Templates.” [Online]. Available: http://www.volere.co.uk/templates.htm. [Accessed: 11-Jun-2014].
[71] A. Moreira, “NFRS in Context.” Lecture presented in Arquiteturas de Software. Universidade Nova de Lisboa, Portugal, 2012.
[72] N. Medvidovic and R. N. Taylor, “Software architecture: foundations, theory, and practice,” *2010 ACM/IEEE 32nd Int. Conf. Softw. Eng.*, vol. 2, pp. 471–472, 2010.
[73] D. Ameller and X. Franch, “Linking quality attributes and constraints with architectural decisions,” *CoRR*, pp. 1–14, 2012.
[74] D. Ameller and X. Franch, “Ontology-based architectural knowledge representation: Structural elements module,” in *Lecture Notes in Business Information Processing*, 2011, vol. 83 LNBIP, pp. 296–301.
[75] D. Ameller, O. Collell, and X. Franch, “ArchiTech: Tool support for NFR-guided architectural decision-making,” *2012 20th IEEE Int. Requir. Eng. Conf.*, pp. 315–316, Sep. 2012.
[76] U. Jensen, *Probabilistic Risk Analysis: Foundations and Methods*, vol. 97, no. 459. Cambridge University, 2002, pp. 925–925.
[77] Y. Asnar, P. Giorgini, and J. Mylopoulos, “Goal-driven risk assessment in requirements engineering,” *Requir. Eng.*, vol. 16, no. 2, pp. 101–116, Sep. 2010.
[78] S. Amber, N. Shawoo, and S. Begum, “Determination of Risk During Requirement Engineering Process,” vol. 3, no. 3, pp. 358–364, 2012.
[79] W. Shenkir and P. Walker, “Enterprise risk management: Tools and techniques for effective implementation,” *Inst. Manag. Accountants*, 2007.
[80] Pressman and Construx.com, “List of potential Risks,” 1997. [Online]. Available: http://groups.engin.umd.umich.edu/CIS/course.des/cis375/projects/risktable/risks.htm. [Accessed: 19-Mar-2014].
[81] University of Edinburgh, “Potencial Sources of Risk.” [Online]. Available: https://www.projects.ed.ac.uk/guidance/risk. [Accessed: 19-Mar-2014].
[82] T. Arnuphaptrairong, “Top Ten Lists of Software Project Risks : Evidence from the Literature Survey,” in *International MultiConference of Engineers and Computer Scientists*, 2011.
[83] K. Boness, A. Finkelstein, and R. Harrison, “A lightweight technique for assessing risks in requirements analysis,” no. 1, pp. 46–57, 2008.
[84] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, “Tropos: An agent-oriented software development methodology,” *Auton. Agent. Multi. Agent. Syst.*, vol. 8, no. 3, pp. 203–236, 2004.
[85] S. L. Cornford, M. S. Feather, V. A. Heron, and J. S. Jenkins, “Fusing quantitative requirements analysis with model-based systems engineering,” in *Proceedings of the IEEE International Conference on Requirements Engineering*, 2006, pp. 272–277.
[86] S. Houmb and F. Den Braber, “Towards a UML profile for model-based risk assessment,” *Crit. Syst. Dev. with UML-Proceedings UML’02 Work.*, 2002.

[87] F. den Braber, I. Hogganvik, M. S. Lund, K. St??len, and F. Vraalsen, “Model-based security analysis in seven steps - A guided tour to the CORAS method,” *BT Technol. J.*, vol. 25, no. 1, pp. 101–117, 2007.
[88] A. Sutcliffe, “Scenario-based requirements analysis,” *Requir. Eng.*, pp. 1–30, 1998.
[89] I. Alexander and N. Maiden, *Scenarios, stories, use cases: through the systems development life-cycle*. West Sussex, England: John Wiley & Sons, 2005.
[90] A. Cockburn, *Writing Effective Use Cases*. Addison-Wesley, 2001.
[91] Cornell University Center for Advanced Computing, “Cornell Virtual Workshop - Use Cases <–> Quality Attribute Scenarios,” 2014. [Online]. Available: http://www.cac.cornell.edu/VW/usecases/comparison.aspx. [Accessed: 19-Jun-2014].
[92] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” 2007.
[93] S. Beecham, T. Hall, and A. Rainer, “Defining a Requirements Process Improvement Model,” *Softw. Qual. J.*, vol. 13, no. 3, pp. 247–279, Sep. 2005.
[94] B. Solemon, S. Shahibuddin, and A. Ghani, “Re-defining the Requirements Engineering Process Improvement Model,” in *APSEC ’09*, 2009.
[95] R. Dadhich and U. Chauhan, “Integrating CMMI Maturity Level 3 in Tradicional Software Development Process,” *IJSEA*, vol. 3, no. 1, pp. 17–26, 2012.
[96] H. F. Ahmad and O. Aoba-ku, “An Empirical Study to Investigate Software Estimation Trend in Organizations,” *Int. Work. Component-Based Softw. Eng.*, 2006.
[97] N. Ehsan, a. Perwaiz, J. Arif, E. Mirza, and a. Ishaque, “CMMI / SPICE based process improvement,” *2010 IEEE Int. Conf. Manag. Innov. Technol.*, pp. 859–862, 2010.
[98] S. M. Hwang and A. Background, “Analysis of Relationship among ISO / IEC 15504 , CMMI and K-model,” *Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. 2009. SNPD ’09. 10th ACIS Int. Conf.*, pp. 306–309, 2009.
[99] I. L. Margarido, R. M. Vidal, and M. Vieira, “Lessons Learnt in the Implementation of CMMI® Maturity Level 5,” *2012 Eighth Int. Conf. Qual. Inf. Commun. Technol.*, pp. 47–56, Sep. 2012.
[100] F. M. G. S. Gonçalves, C. I. M. Bezerra, A. D. Belchior, C. C. Coelho, and C. G. S. Pires, “A Strategy for Identifying, Classifying and Prioritizing Improvement and Innovation Actions: A CMMI Level 5 and Six Sigma Approach,” in *Proceedings of the 19th Australian Conference on Software Engineering*, 2008, pp. 104–111.
[101] E. C. Kosaraju, “Compatibility Between Scrum And CMMI: A Study.”
[102] C. R. Jakobsen and K. A. Johnson, “Mature Agile with a Twist of CMMI,” in *Proceedings of the Agile 2008*, 2008, pp. 212–217.
[103] A. M. L. De Vasconcelos, J. Luis, D. Vara, and J. Sánchez, “Towards CMMI-Compliant Business Process-Driven Requirements Engineering,” in *QUATIC*, 2012.
[104] R. Cerón, J. C. Dueñas, E. Serrano, and R. Capilla, “A Meta-model for Requirements Engineering in System Family Context for Software Process Improvement Using CMMI,” in *Proceedings of the 6th International Conference on Product Focused Software Process Improvement*, 2005, pp. 173–188.
[105] S. Jan and M. N. Majeed, “Role of Requirement Management as a CMMI Process Area in Requirement Engineering,” *Int. J. Comput. Appl.*, vol. 51, no. 19, pp. 25–28, 2012.
[106] A. M. L. de Vasconcelos, G. Giachetti, B. Marín, and O. Pastor, “Towards a CMMI-Compliant Goal-Oriented Software Process through Model-Driven Development,” in *The Practice of Enterprise Modeling*, P. Johannesso, J. Krogstie, and A. L. Opdahl, Eds. Springer Berlin Heidelberg, 2011, pp. 253–267.
[107] F. J. Wang, C.H., and Wang, “A Model for Achieving the Goals of Requirement Development PA in CMMI Level 3 on Workflow Applications,” 2006.
[108] I. Lavi, S. Aserraf, and A. Modai, “Mistakes , Obstacles and Conflicts in using CMMI for Process Improvement,” in *SEPG North America*, 2008.
[109] E. Yoshidome, R. Maurício, and W. Lira, “Apoio Sistematizado à Implementação do Processo de Desenvolvimento de Requisitos do MPS. BR e CMMI a partir do Uso de Ferramentas de Software Livre.,” in *WER*, 2012.
[110] M. Tarnowski, “CMMI contellations,” *Plays-In-Business*, 2010. [Online]. Available: http://plays-in-business.com/2010/12/cmmi-overview-1/?lang=en.

Appendix A

Relationship between CMMI Levels

Table I - Relationship between capability and maturity levels, taken from [27]

Name	*Abbr.*	*ML*	*CL1*	*CL2*	*CL3*
Configuration Management	CM	2			
Measurement and Analysis	MA	2			
Project Monitoring and Control	PMC	2			
Project Planning	PP	2	**Target profile 2**		
Process and Product Quality Assurance	PPQA	2			
Requirements Management	REQM	2			
Supplier Agreement Management	SAM	2			
Decision Analysis and Resolution	DAR	3			
Integrated Project Management	IPM	3			
Organizational Process Definition	OPD	3			
Organizational Process Focus	OPF	3			
Organizational Training	OT	3			
Product Integration	PI	3			
Requirements Development	RD	3	**Target profile 3**		
Risk Management	RSKM	3			
Technical Solution	TS	3			
Validation	VAL	3			
Verification	VER	3			
Organizational Process Performance	OPP	4	**Target profile 4**		
Quantitative Project Management	QPM	4			
Causal Analysis and Resolution	CAR	5	**Target profile 5**		
Organizational Performance Management	OPM	5			

Appendix B

Survey performed at CMMI Portugal conference

Assessment of the CMMI-DEV 1.3 process in the Software Industry

In the context of my MSc dissertation, I am evaluating the most common methods and tools for Requirements Management and Development in industry, a process area required by CMMI-DEV maturity level 3. This survey will also help me to ascertain the relationship between the methods and processes used in Requirements Development and the CMMI model.

The total anonymity of the sources of this survey will be ensured.

*Mandatory

Affiliation *

Nationality of the organisation

Is your organisation CMMI certified? *

- ○ No
- ○ Yes, appraised at maturity level 2
- ○ Yes, appraised at maturity level 3
- ○ Yes, appraised at maturity level 4
- ○ Yes, appraised at maturity level 5

1. Which techniques are used in your organisation to identify the stakeholder needs? *

Stakeholder Needs = the gathered high-level requirements from which the customer requirements are derived.

- ☐ Interviews
- ☐ Workshops
- ☐ Scenarios/Storyboards
- ☐ Design Thinking
- ☐ Prototypes
- ☐ Document analysis
- ☐ Brainstorming
- ☐ Shadowing / Observational methods
- ☐ Requirements reuse
- ☐ None
- ☐ Other:

(You can choose more than one option)

2. Which techniques are used in your organisation to model the customer requirements? *

Customer Requirements = the documented translation of stakeholder needs (and expectations), which may be product characteristics or quality of service.

- ☐ Natural Language
- ☐ Goal-Oriented (e.g., i*)
- ☐ Viewpoints
- ☐ Business Process Modelling (e.g., BPMN)
- ☐ UML (e.g., Use Cases, Activity Diagrams)
- ☐ None
- ☐ Other:

(You can choose more than one option)

3. Which techniques are used in your organisation to model the product requirements? *

Product Requirements = the refined customer requirements into a more technical, precise and detailed set. These can be used for design decisions.

- ☐ Natural Language
- ☐ Data-Flow Modelling (e.g., SSADM, SADT, DeMarco)
- ☐ Object-Oriented Approaches (e.g., UML)
- ☐ Goal-Oriented (Volere, KAOS, i*, NFR Framework)
- ☐ Problem-Oriented (e.g., Jackson Problem Frames)
- ☐ Aspect-Oriented (e.g., Theme/Doc, Arcade)
- ☐ Use Cases / Scenarios
- ☐ Formal Methods (e.g., Z, VDM)
- ☐ Viewpoint-Oriented (e.g., VORD)
- ☐ None
- ☐ Other:

(You can choose more than one option)

4. Which requirements validation methods are used in your organization? *

- ☐ Inspections (e.g., Fagan Inspections)
- ☐ Walkthroughs
- ☐ Requirements Testing
- ☐ Model-based (formal) Verification and Validation
- ☐ Review checklists
- ☐ Scenarios/User Stories
- ☐ Prototyping
- ☐ None
- ☐ Other:

(You can choose more than one option)

5. Which techniques are used in your organisation to develop Non-Functional Requirements (NFRs)? *

NFRs = requirements that specify how a system is supposed to be, in contrast with functional requirements that define what a system is supposed to do. Examples of NFRs are the security, usability or the performance of a system.

- ☐ Natural Language
- ☐ VORD
- ☐ QFD
- ☐ QARCC
- ☐ Goal driven use cases
- ☐ NFR Framework (Softgoals)
- ☐ Misuse case
- ☐ WinWin
- ☐ MQA
- ☐ None
- ☐ Other:

(You can choose more than one option)

6. Which tools does your organisation use for requirements development? *

- ☐ Visual Studio
- ☐ ReqPro
- ☐ CORE
- ☐ TestLink
- ☐ CASE tools (e.g., MagicDraw, ArgoUML, Enterprise Architect)
- ☐ MS Office (or Open Office, ...)
- ☐ Workspace.com
- ☐ None
- ☐ Other:

(You can choose more than one option)

7. If you were responsible for the Requirements Development Process definition of your organisation, which are the main problems you can report?

You can also state the current problems faced nowadays when developing requirements

Suggestions

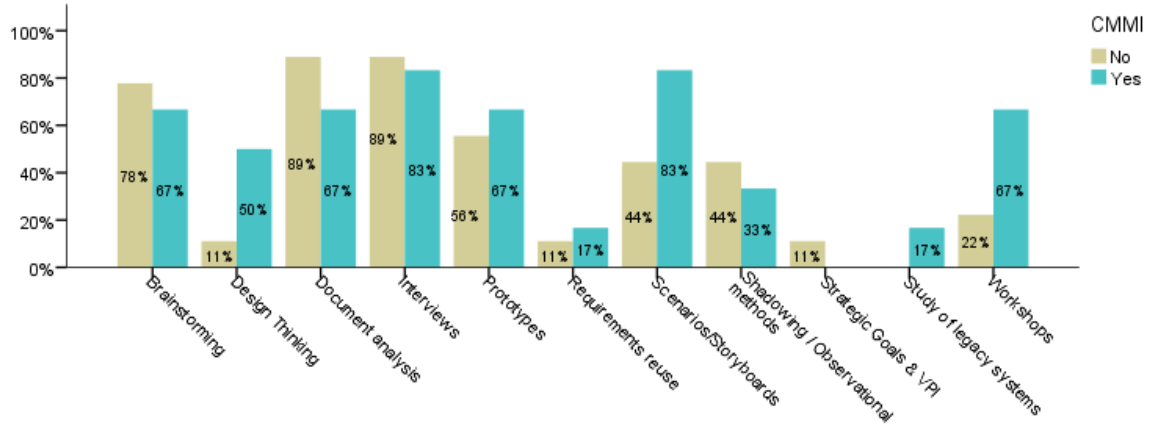
If you have any additional comments concerning the Requirements Development process, please leave your comments.

Interested in the results?

If you are interested in getting a copy of this survey results, please leave your contact.

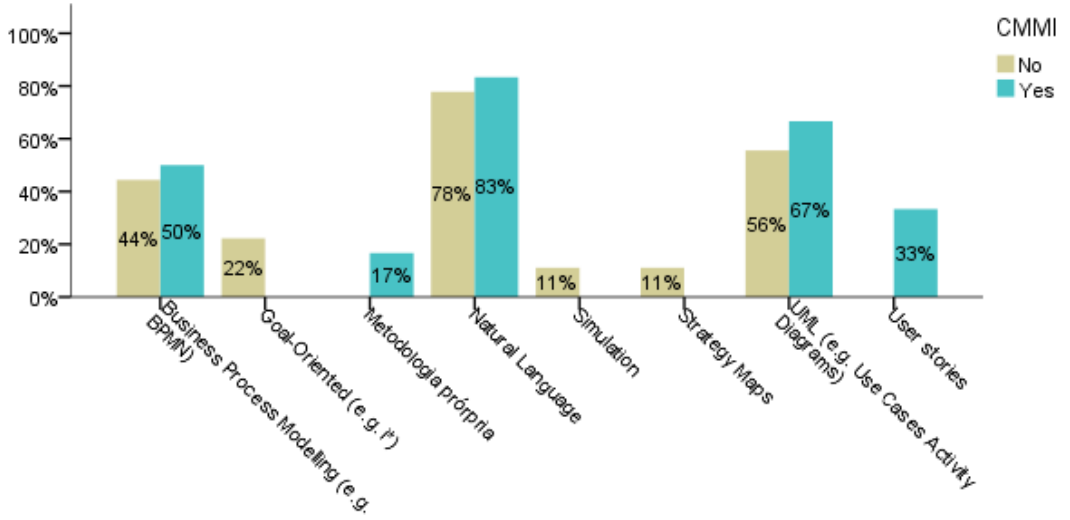
Results

1. *Which techniques are used in your organization to identify the stakeholder needs?*



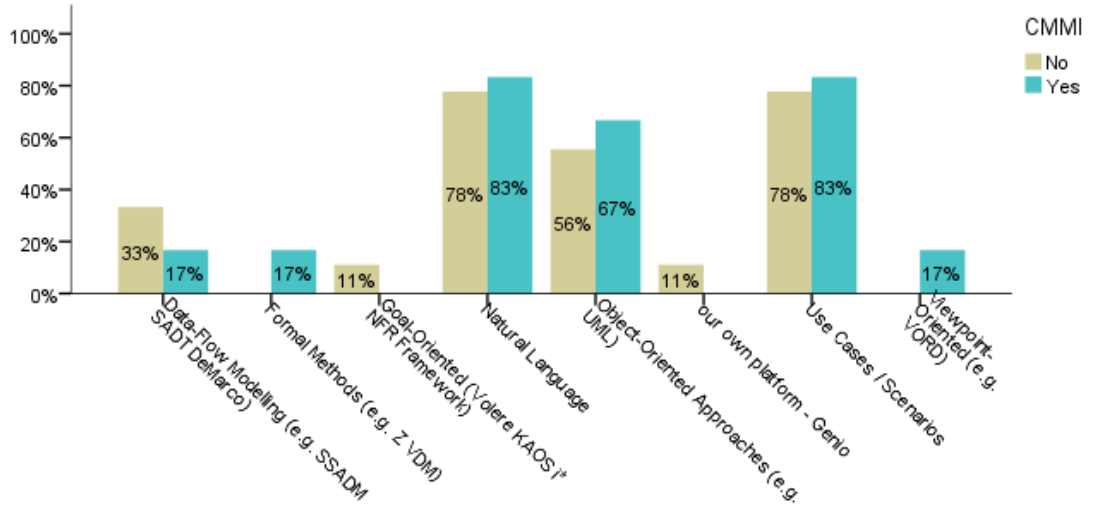
Considering the elicitation techniques, the usage of interviews and document analysis are the most common techniques among both kinds of organizations. Prototypes are also fairly accepted. It is interesting to note that the elicitation of requirements using workshops, design thinking, scenarios and storyboards is more popular for organizations CMMI certified.

2. *Which techniques are used in your organization to model the customer requirements?*



Regarding the modeling techniques for Customer Requirements, the analysis evidence that modeling customer requirements using the natural language seems to be the top choice among organizations. The usage of UML is also well accepted to support the natural language. In addition, User Stories were a technique only reported by certified organizations. Inversely, goal-oriented techniques were barely mentioned and seem to be used only by non-certified companies.

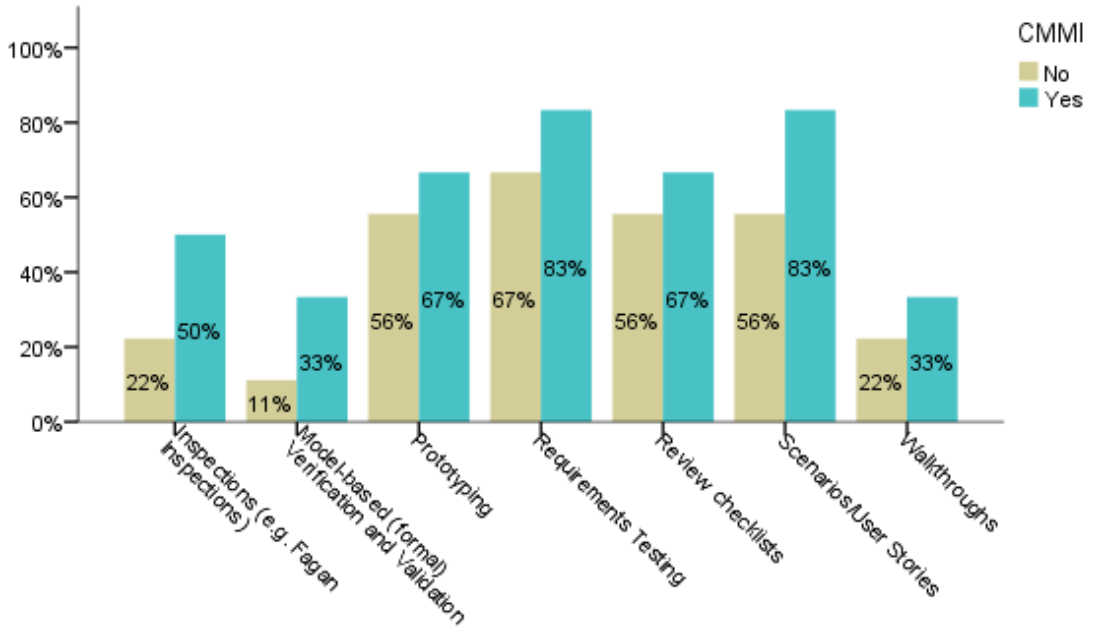
3. *Which techniques are used in your organization to model the product requirements?*



Product Requirements Modeling Technique

Analyzing the answers regarding product requirements modeling, we found almost all of the respondents from all organizations indicated that they modeled product requirements using natural language and Use Cases/Scenarios. Additionally, Object-Oriented Approaches (like UML), also seem to be well-accepted by both certified and non-certified companies. In contrast, viewpoint-oriented, goal-oriented techniques or formal methods were hardly ever mentioned.

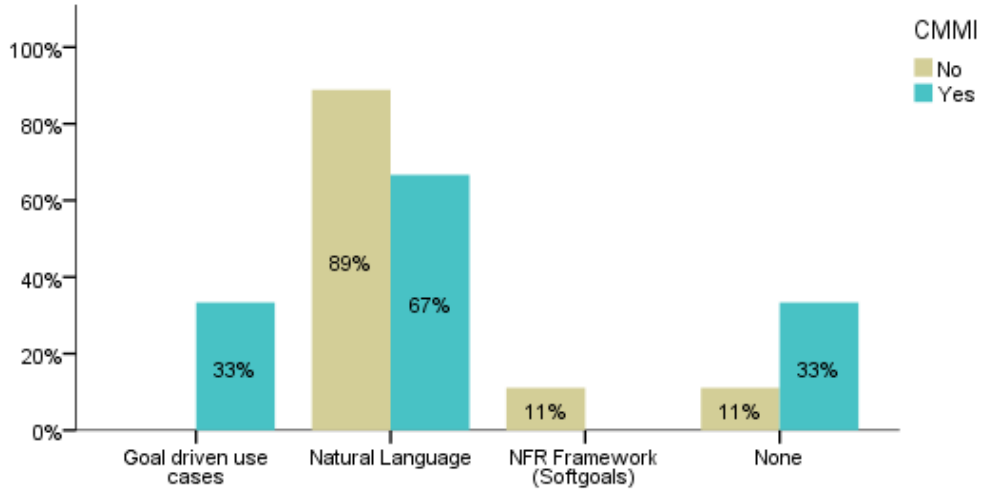
4. *Which requirements validation methods are used in your organization?*



Validation Technique

Considering the requirements validation techniques, the numbers indicate that most certified organizations validate their requirements using a combinations of techniques, particularly using requirements testing and Scenarios or User Stories. It is also interesting to note that formal validation techniques, like inspections and model-based V&V, are more popular among certified organizations.

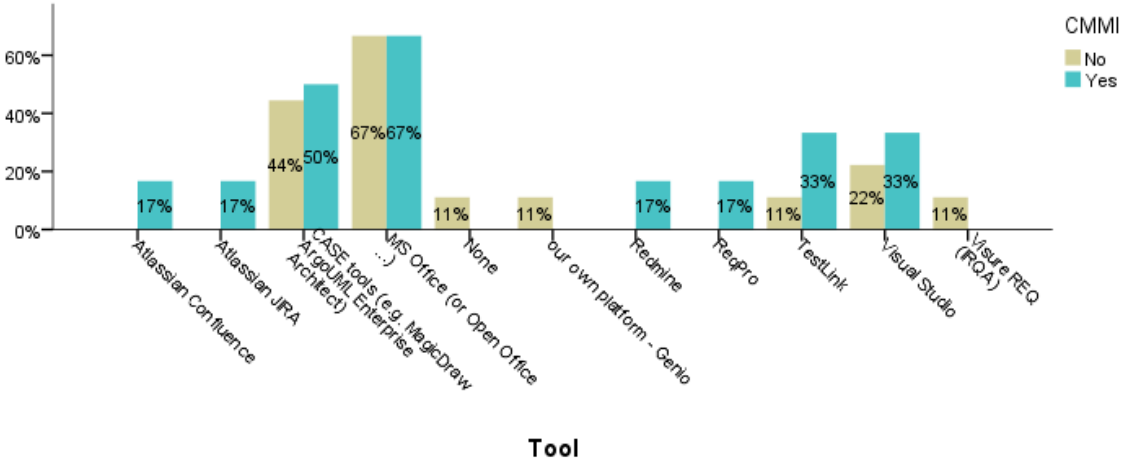
5. *Which techniques are used in your organization to develop Non-Functional Requirements (NFRs)?*



Non-Functional Requirements Modeling Technique

Analyzing the answers regarding Non-Functional Requirements modeling, it is clear that the usage of natural language is the preferred technique. Moreover, goal-driven use cases were also a technique indicted by 33% of the respondents from certified organizations. Nevertheless, some certified and non-certified organizations stated that still don't address this kind of requirements.

6. *Which tools does your organization use for requirements development?*



Tool

Concerning the tools used to support the modeling activities, MS Office was the most reported tool by both certified and non-certified organizations. Case Tools to specify and model requirements also seem to be well-accepted by both organizations. Conversely, requirements management tools, like RedMine, ReqPro, JIRA or TestLink, were mentioned mostly by certified organizations.

Appendix C

CMMI Gap Analysis of the current process

Table II - PII table of Requirements Management process area

Goals	Practices	Sub-Practices	Rating	Objective Evidence
G1 Manage Requirements	1.1 Understand Requirements	1. Establish criteria for distinguishing appropriate requirements providers.	±	Stakeholders are identified in the project plans; they are no clearly stated in the requirements documents. The criteria are ask the customer
		2. Establish objective criteria for the evaluation and acceptance of requirements.	✓	Requirements are registered on TestLink tool, along with their test cases
		3. Analyze requirements to ensure that the established criteria are met.	✓	Done in activity 2 of REQM process and again after the refinement of the requirements during a review meeting in the design phase
		4. Reach an understanding of the requirements with the requirements provider so that the project participants can commit to them.	✓	Gate 2 and 3 of the project planning process
	1.2 Obtain Commitment to Requirements	1. Assess the impact of requirements on existing commitments.	✓	Impact study of change requests
		2. Negotiate and record commitments.	✓	Meetings with the project team and customer are conducted to obtain commitment to the project plans. Formal customer approval of requirements
	1.3 Manage Requirements Changes	1. Document all requirements and requirements changes that are given to or generated by the project.	✓	Requirements are documented in TestLink and in requirements specification documents
		2. Maintain a requirements change history, including the rationale for changes.	✓	Changes are formally proposed in change requests templates
		3. Evaluate the impact of requirement changes from the standpoint of relevant stakeholders.	✓	Changes are evaluated (following a Risk Management process) and later formally approved by the project manager
		4. Make requirements and change data available to the project.	✓	Documentation in the Knowledge Tree tool
	1.4 Maintain Bidirectional Traceability of Requirements	1. Maintain requirements traceability to ensure that the source of lower level (i.e., derived) requirements is documented.	✓	TestLink tool is used to record the requirements and its dependencies, generating a traceability matrix of dependencies among requirements and between requirements and test cases
		2. Maintain requirements traceability from a requirement to its derived requirements and allocation to work products.	✓	
		3. Generate the requirements traceability matrix.	✓	
	1.5 Ensure Alignment	1. Review project plans, activities, and work products for consistency with	±	Peer reviews at the end of each phase of the execution

	Between Project Work and Requirements	requirements and changes made to them.		process are not formally conducted
		2. Identify the source of the inconsistency (if any).	±	Peer reviews at the end of each phase of the execution Process are not formally conducted
		3. Identify any changes that should be made to plans and work products resulting from changes to the requirements baseline.	±	Peer reviews at the end of each phase of the execution process are not formally conducted
		4. Initiate any necessary corrective actions.	±	each of these peer reviews activity would be followed by a resolving issues activity

Table III - PII table of Requirements Development process area

Goals	*Practices*	*Sub-Practices*	*Rating*	*Objective Evidence*
G1 Develop Customer Requirements	1.1 Elicitation of Needs	1. Engage relevant stakeholders using methods for eliciting needs, expectations, constraints, and external interfaces.	✓	The Business Manager elicits requirements with the customer in their first meeting
	1.2 Transformation of needs into Customer Requirements	1. Translate stakeholder needs, expectations, constraints, and interfaces into documented customer requirements.	✓	Requirements are translated into the documented Customer Requirements List included in the PMP
		2. Establish and maintain a prioritization of customer functional and quality attribute requirements.	×	Requirements are not prioritized. But they are grouped according to deliveries
		3. Define constraints for verification and validation.	✓	Assumptions and constraints in the PMP
G2 Develop Product Requirements	2.1 Establishment of Product and Product-component requirements	1. Develop requirements in technical terms necessary for product and product component design.	✓	Requirements Specification activity of REQM process, using a technical design specification template
		2. Derive requirements that result from design decisions.	±	Rationale behind those decisions is not registered and the review of the requirements after the technical design specification is unstated
		3. Develop architectural requirements capturing critical quality attributes and quality attribute measures necessary for establishing the product architecture and design.	×	Quality attributes are not captured systematically and they don't directly drive the design decisions of the project
		4. Establish and maintain relationships between requirements for consideration during change management and requirements allocation.	✓	Requirements are registered in TestLink, along with their dependencies
	2.2 Allocation of Product Component Requirements	1. Allocate requirements to functions.	✓	Requirements are grouped according to modules in TestLink
		2. Allocate requirements to product components and the architecture.	✓	Requirements are allocated to the architecture in the technical design specification
		3. Allocate design constraints to product components and the architecture.	×	The design constrains are currently blended with requirements, so subpractice 3 is not met
		4. Allocate requirements to delivery increments.	✓	Done during the analysis of the requirements list for the PMP

		5. Document relationships among allocated requirements.	✓	Aligned with subpractice 4 of SP 2.1
	2.3 Identification of Interface Requirements	1. Identify interfaces both external to the product and internal to the product (i.e., between functional partitions or objects).	±	Only the external interfaces are registered on TestLink
		2. Develop the requirements for the identified interfaces.	✓	Interface requirements registered on TestLink
G3 Analyze and Validate Requirements	3.1 Establishment of operational concepts and scenarios	1. Develop operational concepts and scenarios that include operations, installation, development, maintenance, support, and disposal as appropriate.	×	No scenarios or operational concepts defined for the projects
		2. Define the environment in which the product or product component will operate, including boundaries and constraints.	±	Suggested in the technical design specification document, but is not systematically done
		3. Review operational concepts and scenarios to refine and discover requirements.	×	Aligned with subpractice 1 of SP 3.1
		4. Develop a detailed operational concept, as products and product components are selected, that defines the interaction of the product, the end user, and the environment, and that satisfies the operational, maintenance, support, and disposal needs.	×	Aligned with subpractice 4 of SP 3.1
	3.2 Establishment of required Functionality and QA	1. Determine key mission and business drivers.	✓	Performed by the BM in the proposal
		2. Identify desirable functionality and quality attributes.	±	Documentation of some quality attributes in some projects. Functional design template with process flows
		3. Determine architecturally significant quality attributes based on key mission and business drivers.	×	Quality attributes don't drive the architecture
		4. Analyze and quantify functionality required by end users.	±	Functional design template whit process flows, inputs, outputs and their responsibilities. Functionality is not measured
		5. Analyze requirements to identify logical or functional partitions (e.g., subfunctions).	✓	Functional design template whit process flows, inputs, outputs and their responsibilities
		6. Partition requirements into groups, based on established criteria (e.g., similar functionality, similar quality attribute requirements, coupling), to facilitate and focus the requirements analysis.	✓	Requirements are grouped in TestLink according to their modules (similar functionality)
		7. Allocate customer requirements to functional partitions, objects, people, or support elements to support the synthesis of solutions.	✓	Requirements are grouped in TestLink according to their modules (similar functionality)
		8. Allocate requirements to functions and subfunctions (or other logical entities).	✓	Requirements are grouped in TestLink according to their modules (similar functionality)
	3.3 Analyze Requirements	1. Analyze stakeholder needs, expectations, constraints, and external interfaces to organize them into related subjects	✓	Analyze requirements list in REQM process. Peer review meetings after requirements specification activities. (Execution process)

		2. Analyze requirements to determine whether they satisfy the objectives of higher level requirements.	✓	Analyze requirements list in REQM process. Peer review meetings after requirements specification activities. (Execution process)
		3. Analyze requirements to ensure that they are complete, feasible, realizable, and verifiable.	✓	Analyze requirements list in REQM process. Peer review meetings after requirements specification activities. (Execution process)
		4. Identify key requirements that have a strong influence on cost, schedule, performance, or risk.	×	Not currently performed by Altran
		5. Identify technical performance measures that will be tracked during the development effort.	×	Not currently performed by Altran
		6. Analyze operational concepts and scenarios to refine the customer needs, constraints, and interfaces and to discover new requirements.	×	Not currently performed by Altran
	3.4 Analyze Requirements to Achieve balance	1. Use proven models, simulations, and prototyping to analyze the balance of stakeholder needs and constraints.	±	Partially accomplished as the prototypes are presented in the documentation. Not negotiated with the customer
		2. Perform a risk assessment on the requirements and definition of required functionality and quality attributes.	×	Risk Analysis is not yet performed specifically for the requirements
		3. Examine product lifecycle concepts for impacts of requirements on risks.	×	Risk Analysis is not yet performed specifically for the requirements
		4. Assess the impact of the architecturally significant quality attribute requirements on the product and product development costs and risks.	±	Performed at a very high level by the Practice Manager when he proposes the solution
	3.5 Validate Requirements	1. Analyze the requirements to determine the risk that the resulting product will not perform appropriately in its intended-use environment.	×	Not currently performed by Altran
		2. Explore the adequacy and completeness of requirements by developing product representations (e.g., prototypes, simulations, models, scenarios, and storyboards) and by obtaining feedback about them from relevant stakeholders.	±	Customer approves the specification document with prototypes, however the feedback received by all the relevant stakeholders is constrained as this approval is often emailed
		3. Assess the design as it matures in the context of the requirements validation environment to identify validation issues and expose unstated needs and customer requirement.	±	The assessment of design as it matures is not done systematically after the meeting with the customer

Appendix D

CMMI Gap Analysis of the proposed process

Goals	Practices	Sub-Practices	Rating	Objective Evidence
G1 Develop Customer Requirements	1.1 Elicitation of Needs	1. Engage relevant stakeholders using methods for eliciting needs, expectations, constraints, and external interfaces.	✓	BM and/or PrM elicit customer needs, expectations and constraints through interviews with the customer and brainstorming. If necessary functional consultants can meet with other relevant stakeholders
	1.2 Transformation of needs into Customer Requirements	1. Translate stakeholder needs, expectations, constraints, and interfaces into documented customer requirements.	✓	Customer Requirements in the format of user stories
		2. Establish and maintain a prioritization of customer functional and quality attribute requirements.	✓	User Stories Prioritized according to MoSCoW or Ranking
		3. Define constraints for verification and validation.	✓	Assumptions and constraints written in the PMP
G2 Develop Product Requirements	2.1 Establishment of Product and Product-component requirements	1. Develop requirements in technical terms necessary for product and product component design.	✓	Product requirements translate customer requirements in technical terms
		2. Derive requirements that result from design decisions.	✓	During the functional analysis, new requirements may be derived which should be added to the Product requirements specification. That is the reason for the parallelism of these activities
		3. Develop architectural requirements capturing critical quality attributes and quality attribute measures necessary for establishing the product architecture and design.	✓	Quality Attributes are documented in the product requirements along with technical requirements. Associated to each quality attribute should be a QA Scenario, which contains a measure of that quality
		4. Establish and maintain relationships between requirements for consideration during change management and requirements allocation.	✓	Traceability through TestLink
	2.2 Allocation of Product Component Requirements	1. Allocate requirements to functions.	±	Requirements are grouped in the document by functional components
		2. Allocate requirements to product components and the architecture.	✓	Requirements are grouped in the document by functional components
		3. Allocate design constraints to product components and the architecture.	✓	The product requirements contain design constraints (as technical requirements)
		4. Allocate requirements to delivery increments.	✓	In the case several deliverables are needed, customer requirements are allocated according to their

				prioritization
		5. Document relationships among allocated requirements.	✓	Traceability between requirements is kept in the template and in TestLink
	2.3 Identification of Interface Requirements	1. Identify interfaces both external to the product and internal to the product (i.e., between functional partitions or objects).	✓	Interface requirements are documented for each component, internal interfaces are specified in a component diagram
		2. Develop the requirements for the identified interfaces.	✓	Interface requirements documented for each component with a proper template
G3 Analyze and Validate Requirements	3.1 Establishment of operational concepts and scenarios	1. Develop operational concepts and scenarios that include operations, installation, development, maintenance, support, and disposal as appropriate.	±	Definition of scenarios for operations during Functional Analysis and of the development, installation, maintenance environments in the technical design specification
		2. Define the environment in which the product or product component will operate, including boundaries and constraints.	✓	Definition of development, installation, maintenance environments in the technical design specification
		3. Review operational concepts and scenarios to refine and discover requirements.	✓	Review of operational concepts and scenarios during the Peer Review
		4. Develop a detailed operational concept, as products and product components are selected, that defines the interaction of the product, the end user, and the environment, and that satisfies the operational, maintenance, support, and disposal needs.	✓	Definition of scenarios and operational concepts during Functional Analysis activity
	3.2 Establishment of required Functionality and QA	1. Determine key mission and business drivers.	✓	Included in the project proposal
		2. Identify desirable functionality and quality attributes.	✓	Functionality is identified through use cases and behavior diagrams. Quality attributes requirements are documented and analyzed with QA Scenarios
		3. Determine architecturally significant quality attributes based on key mission and business drivers.	±	Architectural significant quality attributes are analyzed respectively to the components they affect
		4. Analyze and quantify functionality required by end users.	✓	Functionality is analyzed through use cases and behavior diagrams. It is quantified with QA measures
		5. Analyze requirements to identify logical or functional partitions (e.g., subfunctions).	✓	Activity or sequence diagrams where inputs/outputs and responsibilities are defined for each function
		6. Partition requirements into groups, based on established criteria (e.g., similar functionality, similar quality attribute requirements, coupling), to facilitate and focus the requirements analysis.	✓	Requirements are grouped in the document by functional components
		7. Allocate customer requirements to functional partitions, objects, people, or support elements to support the synthesis of solutions.	±	Customer requirements are not directly allocated, but it is possible through traceability
		8. Allocate requirements to functions and subfunctions (or other logical entities).	±	Requirements are grouped in the document by functional components (as other logical entity)

	3.3 Analyze Requirements	1. Analyze stakeholder needs, expectations, constraints, and external interfaces to organize them into related subjects	✓	Mapping of customer requirements to solutions requirements and their components
		2. Analyze requirements to determine whether they satisfy the objectives of higher level requirements.	✓	During the Peer Review meeting, it is analyzed if Solution Req. satisfy the Customer Requirements
		3. Analyze requirements to ensure that they are complete, feasible, realizable, and verifiable.	✓	Performed in the Peer Review meeting with the help of a Peer Review Checklist
		4. Identify key requirements that have a strong influence on cost, schedule, performance, or risk.	✓	Requirements Risk analysis performed for the Customer Requirements, along with their prioritization
		5. Identify technical performance measures that will be tracked during the development effort.	✓	Measures identified in QA Scenarios
		6. Analyze operational concepts and scenarios to refine the customer needs, constraints, and interfaces and to discover new requirements.	✓	Operational concepts and scenarios analyzed during the Peer Review
	3.4 Analyze Requirements to Achieve balance	1. Use proven models, simulations, and prototyping to analyze the balance of stakeholder needs and constraints.	✓	Definition of GUI prototypes or other more sophisticated prototypes depending on what was agreed with the customer
		2. Perform a risk assessment on the requirements and definition of required functionality and quality attributes.	✓	Requirements Risk analysis performed first during Customer Requirements definition and them revised in the peer review and management of changes
		3. Examine product lifecycle concepts for impacts of requirements on risks.	×	
		4. Assess the impact of the architecturally significant quality attribute requirements on the product and product development costs and risks.	✓	Quality Attributes trade-off analysis performed for architectural relevant Quality Attributes, requirements risks for Quality attributes
	3.5 Validate Requirements	1. Analyze the requirements to determine the risk that the resulting product will not perform appropriately in its intended-use environment.	±	Requirements are validated with the customer, but validation risks are not formally addressed
		2. Explore the adequacy and completeness of requirements by developing product representations (e.g., prototypes, simulations, models, scenarios, and storyboards) and by obtaining feedback about them from relevant stakeholders.	✓	Requirements Validation with the customer includes the presentation of prototypes that can be simple interface mockups or other more sophisticated prototypes depending on the customer/project
		3. Assess the design as it matures in the context of the requirements validation environment to identify validation issues and expose unstated needs and customer requirement.	±	PM validates artifacts created by consultants throughout the several stages of the process. Final Peer Review. Validation issues are not formally considered

Appendix E

Validation Survey (used for Altran's Experts)

New Requirements Management and Development Process

1. O processo apresentado é:
   - ☐ Claro
   - ☐ Simples
   - ☐ Completo
   - ☐ Realista
   - ☐ Pouco claro
   - ☐ Complexo
   - ☐ Pouco completo
   - ☐ Idealista

2. As seguintes definições/utilizações dos conceitos NÃO foram claras:
   - ☐ Stakeholders' Needs
   - ☐ Business Requirements
   - ☐ Customer Requirements
   - ☐ Solution Requirements
   - ☐ Components
   - ☐ Quality Attributes

3. Concorda com a utilização de User Stories para descrever os Requisitos do Cliente?
   - ◯ Sim
   - ◯ Não

4. Utilizaria outros métodos de prioritização em detrimento dos métodos sugeridos?
Por exemplo: AHP, Planning Game, Cumulative Voting

5. Qual a importância de cada actividade no processo?

	Muito Importante	Importante	Indiferente	Pouco importante	Nada Importante
Prioritize Customer Requirements	◯	◯	◯	◯	◯
Select Reusable Requirements	◯	◯	◯	◯	◯
Specify Solution Requirements	◯	◯	◯	◯	◯
Perform Functional Analysis	◯	◯	◯	◯	◯
Review Requirements	◯	◯	◯	◯	◯

6. O que mudaria no processo apresentado?
Adicionaria ou removeria alguma atividade? mudaria alguma responsabilidade, inputs ou outputs?

7. Concorda com a utilização de um único documento para registar os requisitos do cliente, solução e a análise funcional?

- Sim
- Não
- Indiferente

8. Estou familiarizado com a utilização das técnicas seguintes:

	Desconheço	Conheço mas nunca usei	Conheço razoavelmente	Conheço e sou experiente
User Stories				
Casos de uso				
Protótipos dos ecrãs				
Diagrama de componentes				
Diagrama de actividades				
Cenários de atributos de qualidade				

9. A checklist sugerida para Peer Review é útil?

	1	2	3	4	5	
Nada útil						Muito útil

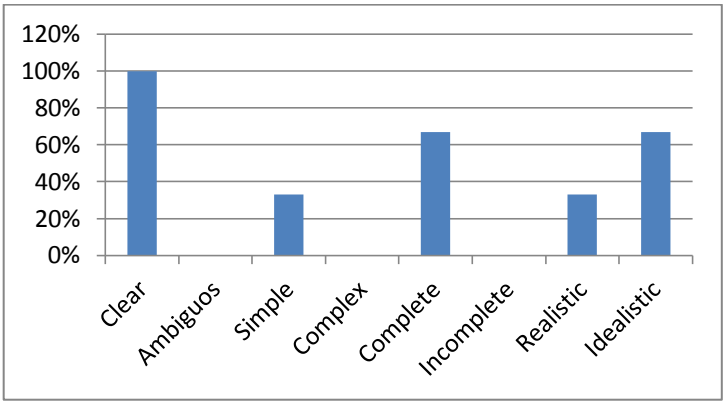
10. Nos projetos que já desenvolveu qual o impacto dos Atributos de Qualidade?

	1	2	3	4	5	
Sem impacto						Grande impacto

11. Sugestões e outras observações:

Results

Q1. The process presented is:



According to the survey the process was perceived as clear by all the respondents. However, 67% believed that, although complete, it is considered idealistic due to the number of new activities introduced in such a short time.

Q2. The following definitions/concepts were not clear:

The Stakeholder's needs and the Business Requirements were considered unclear by 67% of the respondents. The other concepts were not mentioned.

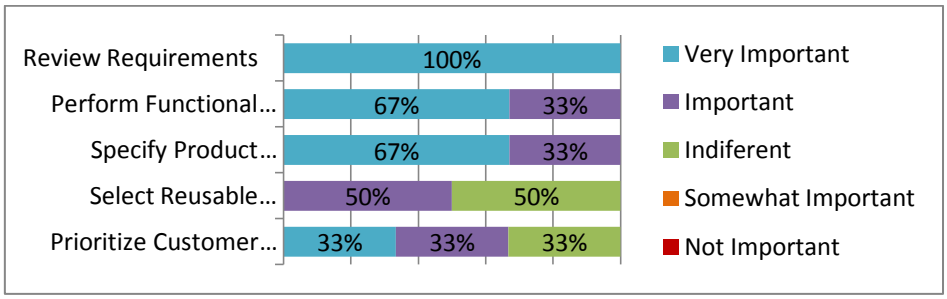
Q3. Do you agree with the use of *User Stories* to describe Customer Requirements?

The use of User Stories was agreed by 67% of the respondents and rejected by the other 33%.

Q4. Would you use other prioritization techniques instead of the ones suggested?

The techniques proposed in the process were accepted by all the respondents.

Q5. What is the importance of each activity in the process?



The peer review was considered a very important activity in the process. The specification of Product Requirements and the Functional analysis activities were considered very important by 67% of the respondents and important by the other 33%. Half of the respondents considered the reusability practices as important in the process. However, the other half believed that they are indifferent.

Regarding the Customer Requirements prioritization, it was equally classified as very important, important and indifferent.

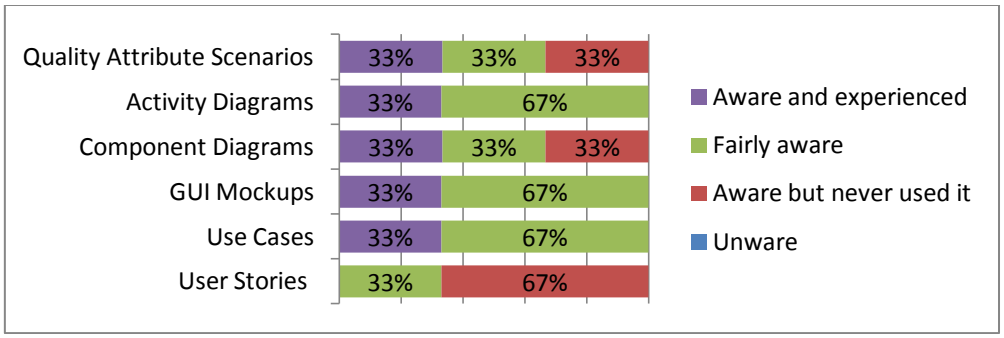
Q6. What would you change in the presented process?

The respondents believe that some changes should be made to the process. In particular they suggested that (1) the order of inputs of the Practice Managers regarding reuse of requirements should happen in an initial phase, before the kick-off; (2) rename the "Product Requirements" for another term, for instances Solution Requirements; (3) open alternative methods that can be described as binding, since some customers may want to do things differently; (4) Functional Analysis starting before the definition of Solution Requirements.

Q7. Do you agree with the use of a single document to register Customer Requirements, Product Requirements and Functional Analysis?

All the respondents agreed with the use of a single document to store all requirements related information.

Q8. I am familiar with the use of the following techniques:



As shown in the Figure, the respondents were aware of all the techniques presented. However, the awareness level varied from *experienced users* to *never applied*. In particular, 67% of the respondents never applied the user stories technique. The activity diagrams, GUI Mockups and the Use Cases seem to be the most proficient techniques. Regarding Quality Attribute Scenarios and components diagrams, the results indicate that some training should be considered for at least 33% of the inexperienced respondents.

Q9. The checklist suggested for the Peer Review is useful?

The Peer Review checklist was considered very useful by 67% of the respondents and useful by the other 33%.

Q10. In the projects you were involved, what was the impact of Quality Attributes?

The Quality Attributes were considered as having a big impact in the projects by 67% of the respondents and a reasonable impact by 33%.

Q11. Suggestions and other observations:

The following suggestions were made by the respondents: (1) apply the changes suggested in the meeting; (2) In the preparation of estimates and commercial proposals should also exist the knowledge of the potential impacts of the presented process, given the impact that these changes might have in the scope, time and cost.

Appendix F
Case Study Application

The Project Requirements Specification document produced from the application of the selected Case Study is presented below.

Project Requirements Specification

TEMPLATE

Project: *AltranREQ*

Information

General

Project Name:	AltranREQ
Document title:	AltranREQ – ProjReqSpec. docx
Reference:	

Distribution list

Name	Function	Copy	Information

Table 4 – Distribution list

Change History

Date	Revision #	Author	Revision Description	Approved
18/08/2014	1.0	Vanessa Ramos	Elaboração do documento	

Table 5 – Change history

Reference Documents

Document	Description	Author
AltranREQ – Especificação de Requisitos		Pedro Furtado
AltranREQ – Modelação de Requisitos		Pedro Furtado

Table 6 – Reference documents

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Não foi encontrada nenhuma entrada do índice de ilustrações.

I INTRODUCTION

1. About This Document

1.1. PURPOSE AND SCOPE

This document is intended for the analysis and definition of the requirements identified by Altran Portugal under the proposal of an appropriate solution for the management of requirements for IT projects, *AltranREQ*.

The overall objective of Altran Portugal in this project is to provide an IT solution to support the management of requirements of software projects, supporting project managers and consultants involved in this context.

This project is performed under the scope of Altran's Academy and has the objective of creating an informatics solution to support the definition and management of project requirements.

2. Project Description

2.1. PROBLEM/NEED

The AltranREQ application will aid the project managers and functional teams, providing them tool support to manage and developed the requirements of Altran's Projects. In addition, …. it will allow future requirements reuse.

2.2. ASSUMPTIONS

This application will be developed with the technologies and methodologies learned during Altran's Academy.

II CUSTOMER REQUIREMENTS

1. Stakeholders' Roles

Brief description of each stakeholder involved in this project.

Project Manager – He is the responsible to manage all the project information and resources, planning it, approve the requirements specified and interact with the client.

Functional Consultant – He is allocated to projects in order to specify their requirements, the system interactions and associated artefacts. Therefore, he is the author of the documentation produced.

Administrator – He is responsible to manage the application settings, the user profiles and their permissions.

Client – As the owner of the project, the client provides the information about the needs and constraints of the project. He needs to be sure that his information will be confidential and only shared with appropriate consultants.

Maintainer – The person responsible for maintaining the application in the future.

2. Customer Requirements Description

List of Customer Requirements that describe both functional and non-functional requirements that reflect stakeholder's needs, expectations and constraints. Priorities were given according to the MoSCoW method: {MUST, SHOULD, COULD, WON'T}.

CR 01-F	*User Story Manage Projects*
Description:	*As a Project Manager,* *I want to manage a list of my assigned projects* *So that I can quickly store their information.*
Priority:	*MUST*

CR 02-F	*User Story Manage Requirements*
Description:	*As a Functional Consultant,* *I need to manage both functional and non-functional requirements of a project,* *so that I can specify the functionalities of the system.*
Priority:	*MUST*

CR 03-F	*User Story Manage Use Cases*
Description:	*As a Functional Consultant,* *I would like to specify the Use Cases of a project,* *So that I can specify the system's interactions.*
Priority:	*MUST*

CR 04-F	*User Story Export Information*
Description:	*As a Project Manager,* *I would like to export all elements related to a project,* *So that I can print them and present them directly to the client.*
Priority:	*SHOULD*

CR 05-F	*User Story Manage Users*
Description:	*As an Administrator of the application,* *I want to manage its users with different permissions,* *So that I can restrict their access according to their functions.*
Priority:	*MUST*

CR 06-F	*User Story Manage Clients*
Description:	*As a Project Manager,* *I would like to manage a list of clients,* *So that I can associate them to projects.*
Priority:	*MUST*

CR 07-F	*User Story Maintain Log of Operations*
Description:	*As an Admin,* *I would like to consult the log of the operations performed in the application, so that I can monitor the actions performed in the system.*

Priority:	*COULD*

CR 08-F	*User Story Maintain Requirements Evolution*
Description:	*As a Functional Consultant,* *I would like to keep an history of the requirements versions,* *So that I can remember why we updated it.*
Priority:	*MUST*

CR 09-F	*User Story Manage Business Categories*
Description:	*As a Project Manager,* *I would like to have a business category associated to requirements,* *so that I can reuse them in future projects.*
Priority:	*COULD*

CR 10-F	*User Story Manage Interactions*
Description:	*As a Project Manager,* *I need to interact with the client through emails, meetings or phone calls, so that I can exchange information with him.*
Priority:	*SHOULD*

CR 01-NF	*User Story Security*
Description:	*As a client,* *I want the application to be used only by legitimate users,* *So that I can ensure my information is secure.*
Priority:	*MUST*

CR 02-NF	*User Story Usability*
Description:	*As a User of the Application,* *I want it to be intuitive and use it without needing a manual,* *So that I can quickly perform the actions I need.*
Priority:	*MUST*

CR 03-NF	*User Story Compatibility*
Description:	*As a User of the Application,* *I would like it to be compatible with several browsers,* *So that I can use the one I'm used to.*
Priority:	*SHOULD*

CR 04-NF	*User Story Interoperability*
Description:	*As an Administrator,* *I would like the application to exchange information with our Personal Management System, so we can have updated information about each consultant.*
Priority:	*COULD*

CR 05-NF	*User Story Maintainability*
Description:	*As a future Maintainer,* *I want the application to be easily maintained,* *So that I can quickly correct defects and accommodate changes.*
Priority:	*SHOULD*

III FUNCTIONAL ANALYSIS

1. Functional description

This chapter describes all the interactions of the product with the environment, end users, and other components for all modes and states within operations, development, deployment, delivery, maintenance, training, and disposal. Scenarios were described with Use Cases and then analysed with sequence diagrams to identify all the sub-functions necessary to the accomplishment of each functionality.

1.1 SCENARIO SPECIFICATION

1.2.1. UC 01 – "Export All Information"

UC 1.	Export all Project Information
Actor:	Administrator, Functional Analyst, Project Manager
Description:	The user exports the integral project information into a MSWord document.
Evaluation Criteria:	A MSWord document with all the project information is successfully downloaded and opened with a MSWord tool.
Precondition:	The user must be logged in the application (UC XX – Login)
Main Scenario:	1. Search Project and/or select project; 2. On the menu select the option "Export"; 3. The system presents a confirmation message; 4. The system exports all the information of the project into a MS Word document. The data to export includes all the project details: project information, functional requirements, non-functional requirements and use cases; 5. A word document with the project information is downloaded.
Extension Scenario 1:	At step 3, if the user cancels the operation: a) The system returns to the project details' view.
Post condition:	--

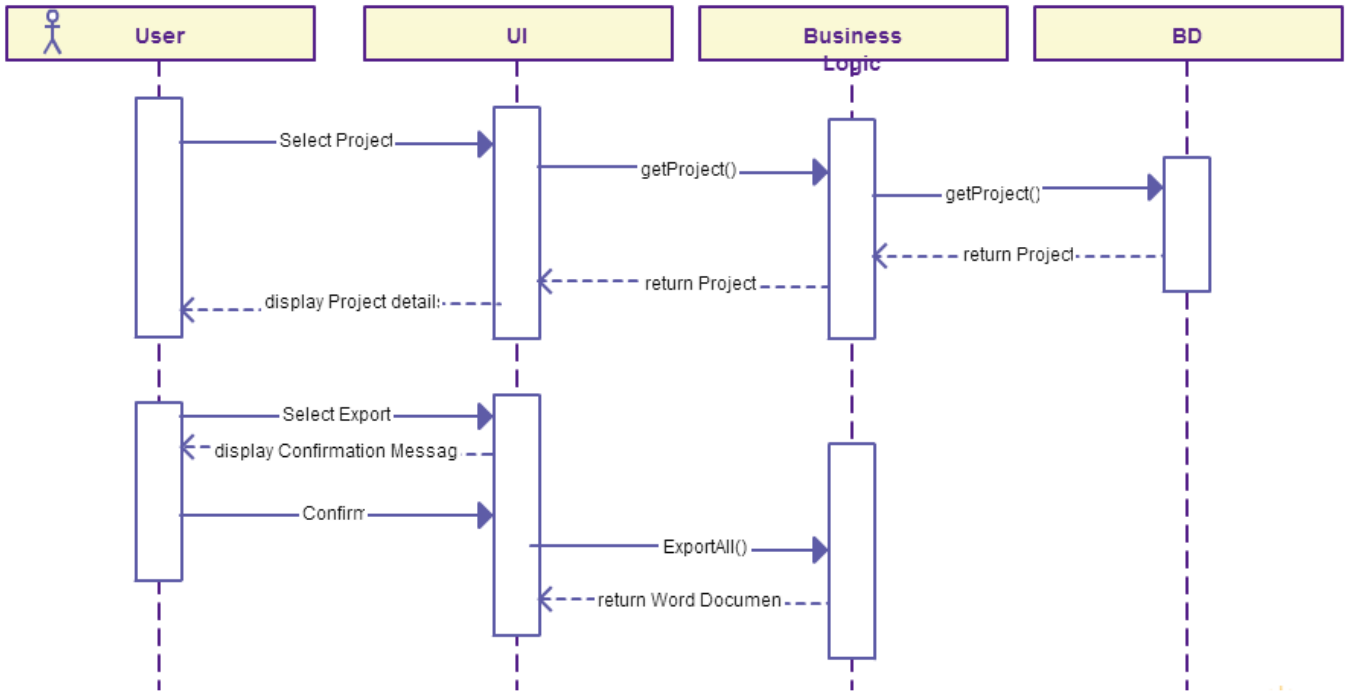


Figure 0.1- Sequence Diagram Export all Information

1.2.2. UC 02 – "Export Functional Requirements"

1.2.3. UC 03 – "Export Use Cases"

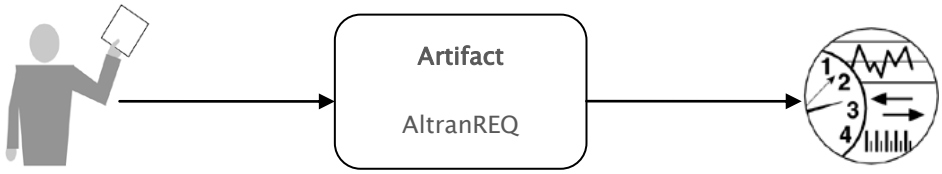
1.2.4. UC 04 – "Export Non-Functional Requirements"

1.2 QUALITY ATTRIBUTE SCENARIOS

For each relevant Quality Attribute identified in chapter III, specify Quality Attribute Scenarios that describe how they can be measured and satisfied.

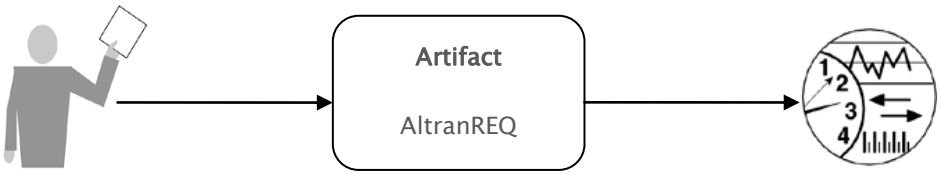
1.1.1. SECURITY

QAS 1. *An unauthorized user attempts to access the system and is blocked.*



Source	Stimulus	Environment	Response	Measure
Unauthorized User	Attempts to access the system	Runtime	Block Access	Access denied successfully

QAS 2. *A legitimate user attempts to access the system and the access is granted with the respective role's rights applied.*



Source	Stimulus	Environment	Response	Measure
Legitimate User	Attempts to access the system	Runtime	Grant Access and role's rights	Access and rights granted successfully

Derived Design Decisions: To achieve these security scenarios, authentication and authorization mechanisms are necessary.

1.1.2. USABILITY

QAS 3. The user learns how to use the application with no manual or training.

Source	Stimulus	Environment	Response	Measure
User	Uses application for the 1st time	Runtime	Uses the application productively	90% of the tasks accomplished with no training

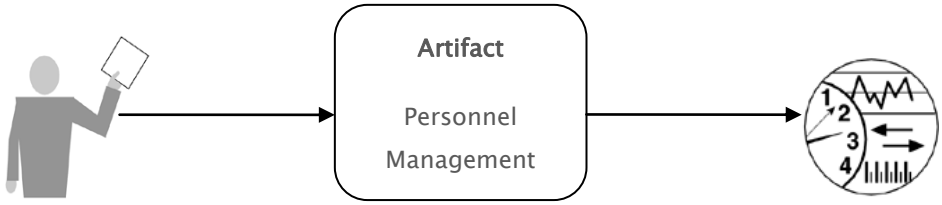
QAS 4. A user performs a removal operation and receives a confirmation message.

Source	Stimulus	Environment	Response	Measure
User	Performs a delete operation	Runtime	Show confirmation message	Zero removal operations with no feedback

Derived Design Decisions: To achieve such usability, standard icons should be used, as well as system feedback on every operation performed, error messages and help in navigation.

1.1.3. INTEROPERABILITY

QAS 5. AtranREQ requests to exchange information about available functional consultants with the Personnel Management System and receives the data correctly 90% of the time.

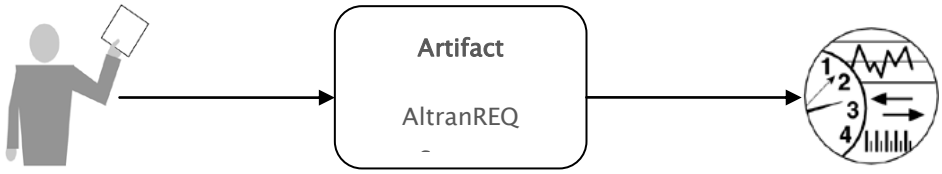


Source	Stimulus	Environment	Response	Measure
AtranREQ	Request to exchange info	Runtime	Exchanged info	Average 90% of info correctly exchanged

Derived Design Decisions: To achieve such interoperability, standard interfaces and communication protocols should be provided.

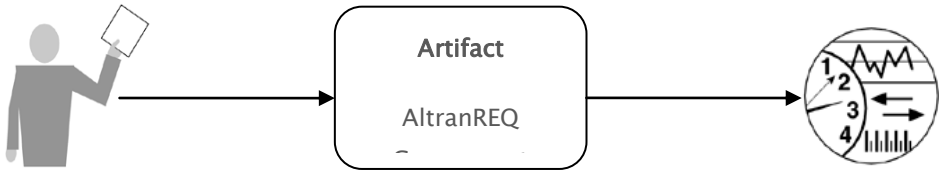
1.1.4. MAINTAINABILITY

QAS 6. The maintainer detects the cause and repairs a defect in less than 5 hours.



Source	Stimulus	Environment	Response	Measure
Maintainer	Repair defect	Repair mode	defect repaired	Less than 5 hours

QAS 7. The maintainer adds a new functionality/ updates an existing one in a given component, introducing zero defects in the remaining components.



Source	Stimulus	Environment	Response	Measure
Maintainer	Add Functionality	Repair mode	Functionality added	Zero defects introduced in other components

Derived Design Decisions: To achieve such maintainability the system should be modular with low coupling between components.

2. Functional Architecture Overview

This chapter provides a high level perspective of the entire system's component and how quality attributes can influence design choices.

2.1. TRADE-OFF ANALYSIS

Impact analysis of the architecturally significant quality attributes on the product.

	Security	Usability	Interoperability	Maintainability
Security		–		
Usability				
Interoperability	–			+
Maintainability				

Justifications:

Measures to prevent authorized access like password and decipher a visually encrypted code improve Security but hurt Usability as they make the system less simple to use.

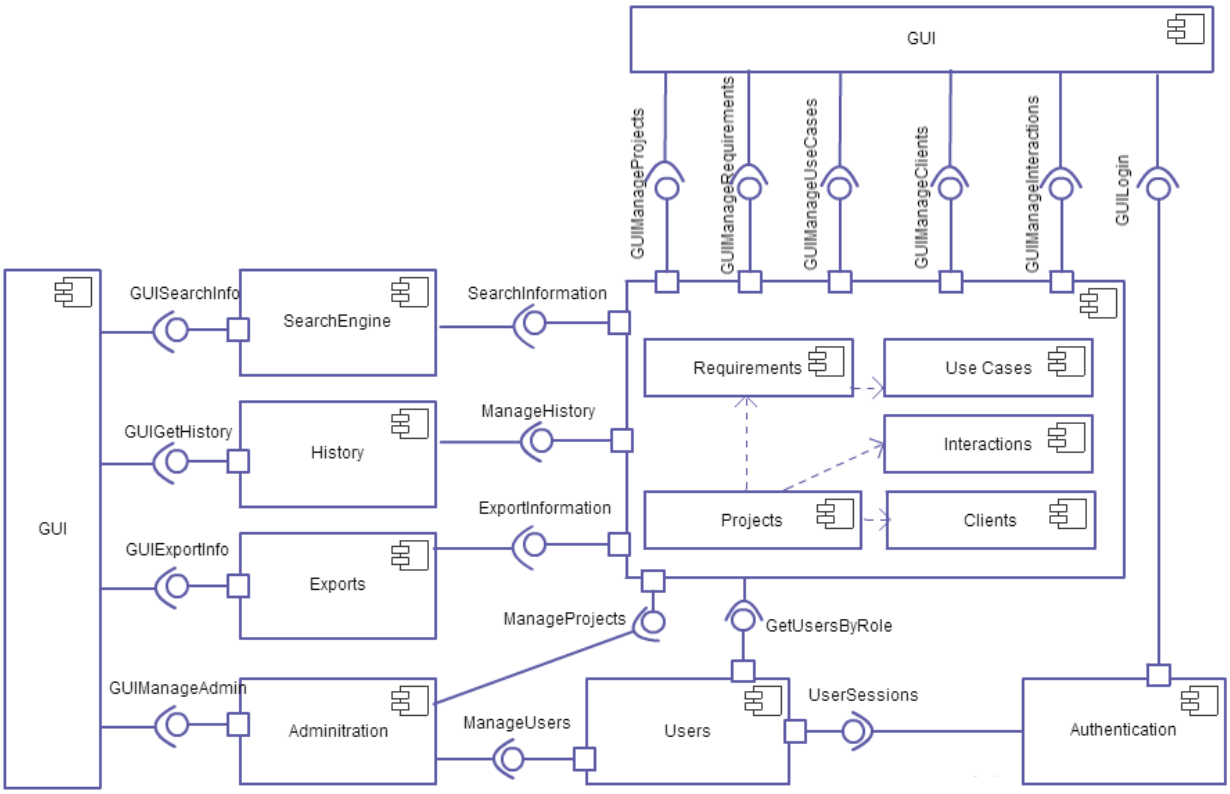
Exchanging information with another system through a web-service is good for interoperability, but might present security threat on the exchanged information. Plus, using standard message formats and communication protocols will improve future maintainability.

	Administration	Exporting	History	SearchEngine	Users	ProjectInfo
Security	X				X	X
Usability				X		X
Interoperability					X	
Maintainability	X					X

The component **ProjectInfo** needs both Security and Usability, which are in conflict. The top priority for users is maximum ease of use, while the top priority for customers is the security of their information. Decisions need to be made about the degree of security and usability needed.
The component **Users** needs to be both Secure and Interoperable. As seen on table X, this two Quality attributes have a negative impact on each other. The user's information needs to be exchanged in a secure way. Design Decisions should balance these two needs.

2.2. COMPONENTS SPECIFICATION

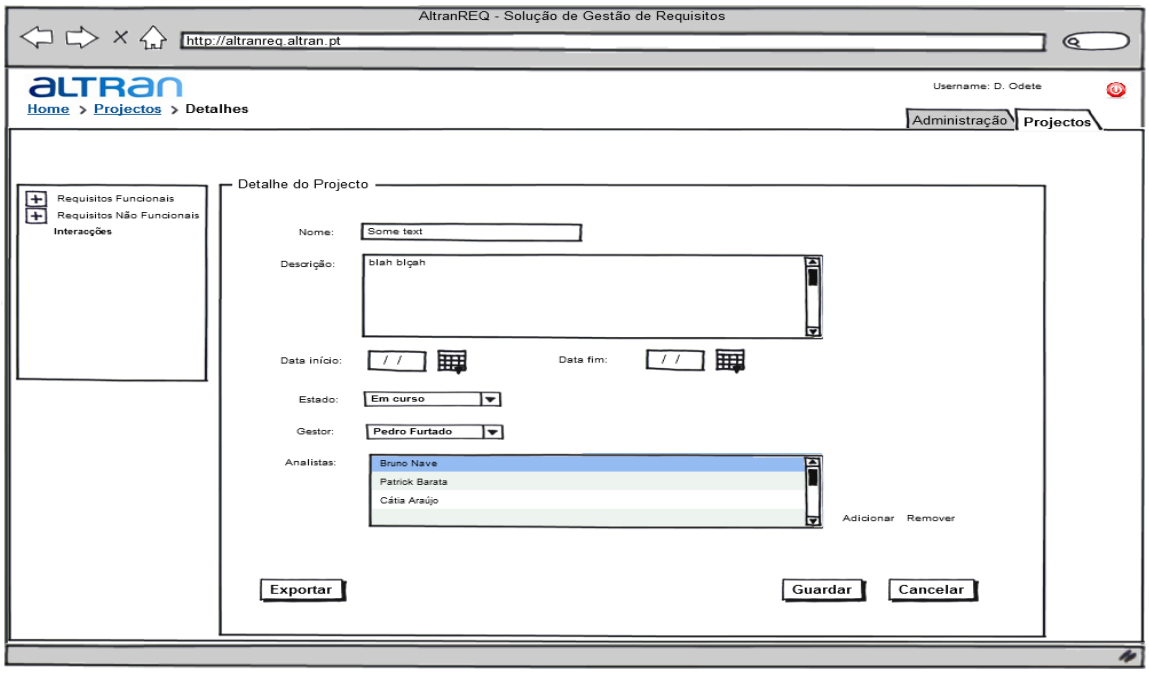
Diagram with the logical architecture of the system:



3. User interface

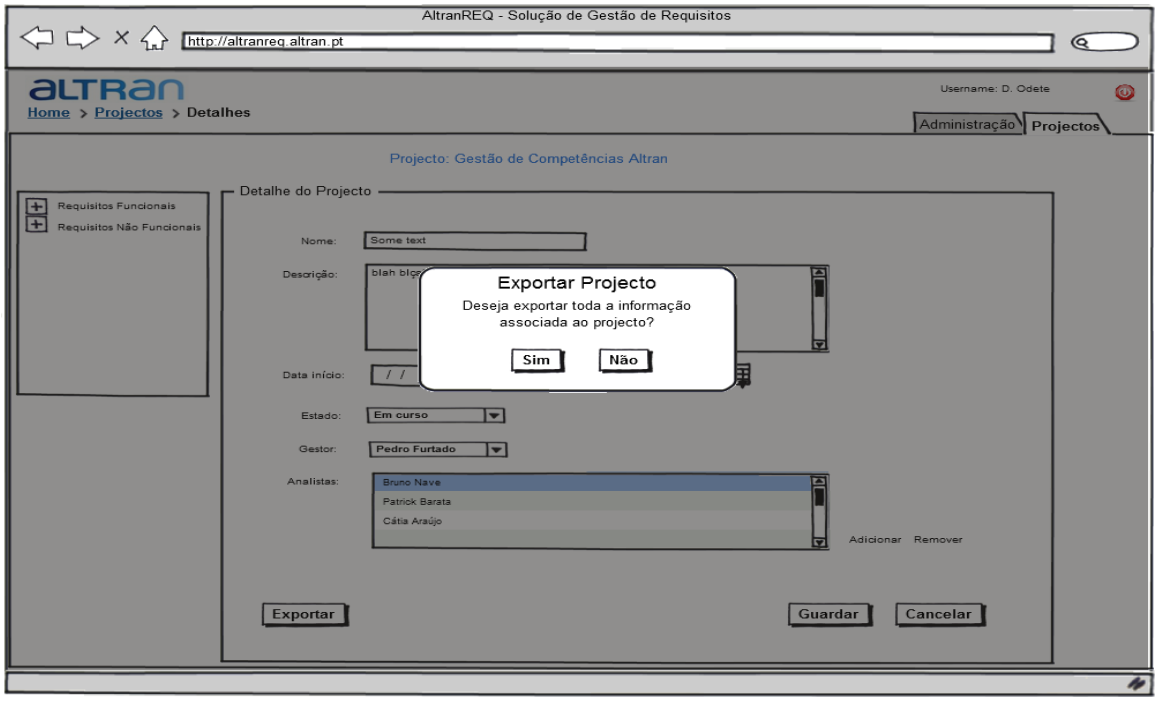
This chapter describes the graphical user interfaces (GUI) of the Presentation Layer components with mock-ups sketches for each distinct screen.

3.1. Data Exporting



SCREEN 1 – Project Details

GUI Object	*Type*	*Action*	*Notes*
Exportar	Button	Go to SCREEN 2 – Export Confirmation	Downloads project information in word document.
Guardar	Button	Go to SCREEN X – <Name>	Save edited project information.
Cancelar	Button	Go to SCREEN Y – <Name>	Cancel changes made on the project info.
Administração	Tab	Go to SCREEN Z – <Name>	Administration operations.



SCREEN 2 - Export Confirmation

GUI Object	*Type*	*Action*	*Notes*
Sim	Button	Go to SCREEN 3 – Generated Document	Downloads project information in word document.
Não	Button	Go to SCREEN 2 – Project Details	Returns to the project details view and doesn't export the data.

SCREEN 3 - Generated Document

4. Third Party Software

The application may interact with the following external software:

- Personnel Management System.
- Tools to generate Word/PDF Documents.

IV SOLUTION REQUIREMENTS

Solution Requirements are a refinement of the Customer Requirements into the developers' language, making implicit requirements into explicit derived requirements. They are the detailed set of requirements of the solution that will satisfy the Customer Requirements.

2. COMP-01: Exporting

2.1. FUNCTIONAL REQUIREMENTS

SR 1.01 - F	*Export Project Information*
Use Case (if available):	UC 01Export All Information, UC 02 Export Functional Requirements, UC 03 Export Use Cases, UC 04 Export Non-Functional Requirements
Mockup (if available):	SCREEN 1 - Project Details
Description:	The application should provide a mechanism to export the selected project information into a printable document: • Use Cases • Functional Requirements • Non-Functional Requirements • Integral Project information
Source:	Altran
Evaluation Criteria:	Obtain a document containing the information selected to export.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

SR 1.02 - F	*Document's Content*
Use Case (if available):	--
Mockup (if available):	SCREEN 3 - Generated Document
Description:	The generated document should be named "*<Project_Name> - Requirements Specification*" and separate Functional Requirements, Non-Functional Requirements and Use Cases in different sections. It also should include an initial informative section containing the following data: • Project Name • Users' Names listed as Authors • Exportation Date • Version

Source:	Altran
Evaluation Criteria:	Obtain a document containing the initial information section filled accordingly.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

2.2. QUALITY ATTRIBUTE REQUIREMENTS

2.2.1. MAINTENANCE

SR 1.03 – QA	*Evolution of Exported Data*
QA Scenario (if available):	QAS 7
Description:	The exportation mechanism should be prepared to accommodate changes in the kinds of information to export, as well as changes in the template's structure and style.
Source:	Altran
Evaluation Criteria:	Changes made and exported successfully.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

2.3. INTERFACE REQUIREMENTS

SR 1.04 – I	*Altran's Template style*
Mockup (if available):	SCREEN 3 – Generated Document
Description:	The generated document should be based on the existing Altran's Requirements Specification Template, following both its structure and graphical styles, including logos, headers, footers, fonts and colors.
Source:	Altran
Evaluation Criteria:	The generated document follows Altran's Requirements Specification Template structure and style.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

2.4. TECHNICAL REQUIREMENTS

SR 1.05 – T	*Exportation Format*

Use Case (if available):	--
Description:	The document can be exported into one of the following formats: • Word • PDF The word document should have the extension ".xdoc", so it is compatible with different word versions.
Source:	Altran
Evaluation Criteria:	The document is exported into word or pdf format, and can is successfully opened with an appropriate tool.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

SR 1.06 - T	*External Export Tools*
Use Case (if available):	--
Description:	This component should be provided with a software tool capable of receiving project's data and generating documents in the appropriate formats.
Source:	Altran
Evaluation Criteria:	
Dependencies:	SR 1.05-T
Size:	*<Estimated Effort>*
History:	Original version 1.0 18/08/2014

3. COMP-02: Specification

…

4. COMP-03: Users

…

5. COMP-04: History

…

6. COMP-05: Administration

…

7. COMP-06: Search Engine

...

8. Crosscutting Solution Requirements

Requirements that are transversal to the application, i.e. those that may apply to several modules/components. Quality Attributes and Technical Requirements are a common type of requirements that can affect several components.

8.1. QUALITY ATTRIBUTE REQUIREMENTS

8.1.1. USABILITY

SR 01 - QA	*Operation's feedback messages*
QA Scenario (if available):	QAS 4
Mockup (if available):	SCREEN 2 - Export Confirmation
Description:	The user should be informed about the state of the actions that he is performing, through feedback messages of the following types: • Error: invalid characters. • Warning: mandatory fields not filled. • Confirmation: request for confirmation of the operations. • Conclusion: notify success/rejection of operations.
Source:	Altran
Evaluation Criteria:	Message display after each transaction performed in the system.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

SR 02 - QA	*Operation's Progress*
QA Scenario (if available):	--
Description:	The solution must provide a mechanism to display the progress of operations that require multiple steps between the system and the user. Example- "Step 2 of 5."
Source:	Altran
Evaluation Criteria:	Display of the current state of the operation.
Dependencies:	N/A

History:	Original version 1.0 18/08/2014

SR 03 – QA	*Navigation Help*
QA Scenario (if available):	--
Description:	The solution should provide a mechanism to aid in the navigation of form fields. This mechanism should allow the navigation among all form fields, as well as provide focus on the fields that need to be filled, or in the first field of the form. Example: Filling the form sequentially with the key "tab".
Source:	Altran
Evaluation Criteria:	Ability to use the mechanism every time it is needed and in any context of the application.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

SR 04 – QA	*User Online Help*
QA Scenario (if available):	QAS 3
Description:	A mechanism that describes the main operations of the application that should be available, in order to help the user understand the context of the tasks he needs. This mechanism should always be present and be specific to the module that the user is using.
Source:	Altran
Evaluation Criteria:	Using the mechanism whenever necessary in any context of the application.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

8.2. INTERFACE REQUIREMENTS

SR 05 – I	*Altran's coorporate identity styles*
Mockup (if available):	SCREEN 1 – Project Details
Description:	The Graphical Interfaces should be based on Altran's corporate identity styles, enriched with a set of colors, shapes and innovative styles, equally coherent with the ones defined in Altran's Corporate style.
Source:	Altran

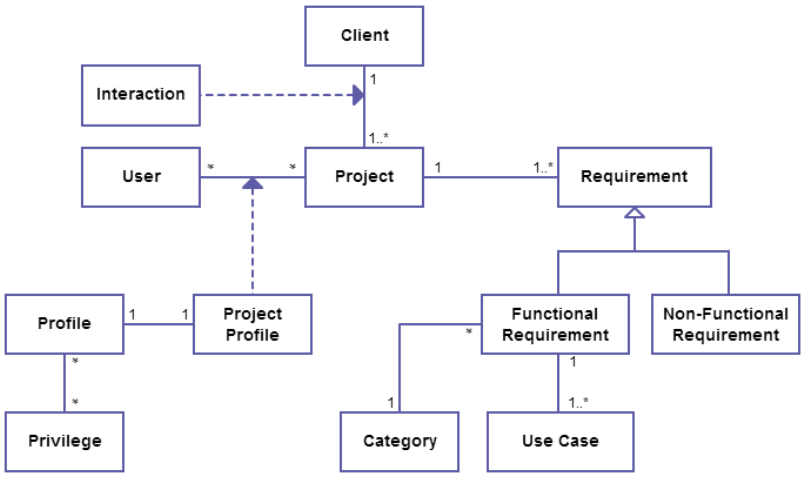
Evaluation Criteria:	Display of Altran's graphical style, including institutional logos.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

8.3. TECHNICAL REQUIREMENTS

SR 06 - T	*Web-based Architecture*
Use Case (if available):	--
Description:	The solution presents an architecture based on a web model, using a client-server component with the application handled through a browser.
Source:	Altran
Evaluation Criteria:	Application handled through a browser.
Dependencies:	N/A
History:	Original version 1.0 18/08/2014

9. Data Requirements

9.1.1. DOMAIN MODEL



9.1.2. SPECIFICATION OF DOMAIN ENTITIES

9.1.2.1. Entity "Project"

Attribute	*Type*	*Mandatory?*	*Visible?*	*Notes*
ID	Number	Y	N	Key
Name	Text	Y	Y	Editable on the screen
Description	Text	Y	Y	Editable on the screen
Start Date	Date	N	Y	Editable on the screen
End Date	Date	N	Y	Editable on the screen
State	List	Y	Y	{OnGoing, Suspended, Closed, Maintenance}
Project Manager	Ref.	N	Y	Ref. to User entity
Terminology	Text	N	Y	Editable on the screen
Client	Ref.	N	Y	Ref. to Client entity
Analysts	RefsList	N	Y	Ref. to User entity
Functional Req.	RefsList	N	Y	Ref. to Functional Req. entity
Non Functional Req.	RefsList	N	Y	Ref. to Non-Functional Req. entity

APPENDIX A
Requirements Traceability

Customer Requirements	*Components*	*Solution Requirements*
CR 01-F	COMP 02	...
CR 02-F	COMP 02	...
CR 03-F	COMP 02	...
CR 04-F	COMP 01	SR 1.01-F
		SR 1.02-F
		SR 1.03-QA
		SR 1.04-I
		SR 1.05-T
		SR 1.06-T
CR 05-F	COMP 05	...
CR 06-F	COMP 02	...
CR 07-F	COMP 04	...
CR 08-F	COMP 04	...
CR 09-F	COMP 02	...
CR 10-F	COMP 02	...
CR 01-NF	System	...
CR 02-NF	System	SR 01 - QA
		SR 02 - QA
		SR 03 - QA
		SR 04 - QA
CR 03-NF	System	...
CR 04-NF	COMP 03	...
CR 05-NF	System	...

INNOVATION MAKERS